R. Joseph Trojan CA Bar No. 137,068
trojan@trojanlawoffices.com
TROJAN LAW OFFICES
9250 Wilshire Blvd., Suite 325
Beverly Hills, CA 90212
Telephone: 310-777-8399
Facsimile: 310-777-8348

Attorneys for Plaintiff,
PHOENIX SOLUTIONS, INC.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

EMC

PHOENIX SOLUTIONS, INC., a California corporation,

Plaintiff,

v.

WELLS FARGO & COMPANY, a Delaware corporation,

Defendant.

CASE NO. CV 08 0863

COMPLAINT FOR DAMAGES AND INJUNCTIVE RELIEF FOR INFRINGEMENT OF U.S. PATENT NOS. 6,633,846, 6,665,640, 7,050,977 AND 7,277,854 UNDER 35 U.S.C. § 271 AND DEMAND FOR JURY TRIAL PURSUANT TO FED. R. CIV. PROC., RULE 38

TROJAN LAW OFFICES
BEVERLY HILLS

//

//

//

//

//

//

//

//

//

Plaintiff, PHOENIX SOLUTIONS, INC. (hereinafter "Plaintiff" or "Phoenix"), hereby complains against Defendant, WELLS FARGO & COMPANY (hereinafter "Defendant" or "Wells Fargo"), as follows:

1. This is a civil action for patent infringement arising under the patent laws of the United States, 35 U.S.C. § 271 *et seq.*

I. THE PARTIES

2. Plaintiff is a corporation organized and existing under the laws of the State of California, with a place of business at 634 Georgia Avenue, Palo Alto, California, 94306.

3. Upon information and belief, Defendant is a corporation organized and existing under the laws of the State of Delaware with a place of business at 420 Montgomery Street, San Francisco, California, 94163.

II. FACTUAL BACKGROUND

4. Plaintiff is the owner by assignment of U.S. Patent Nos. 6,633,846, 6,665,640, 7,050,977, and 7,277,854 (hereinafter "Patents in Suit") directed to "speech recognition software".

5. Plaintiff Phoenix developed the next generation of speech recognition systems that give users the ability to have a verbal conversation with a computer about a subject on which the computer has been programmed to process and generate intelligent responses. One of the first applications of this new technology was its use in telephone customer service lines where the customer calls a computer and a "virtual customer service agent" answers the line and interacts with a caller using "natural speech" akin to a live person.

6. Phoenix encompasses the life work of a pioneer in the field of computer-based speech recognition, Dr. Ian Bennett. Originally from Jamaica, Dr. Bennett graduated with honors from the University of British Columbia and went on to receive his Master's and Doctorate degrees in electrical engineering from

Stanford University. While at Stanford, Dr. Bennett developed the first practical analog processor for speech compression. After graduation he held technical engineering positions with several high technology companies and contributed to device and product development. As a consultant to the Variable Speech Corp. of Tokyo, Japan, he contributed to the development of an analog speech compression VLSI speech processor used for audio compression in consumer speech recorders. In 1994, Dr. Bennett began the development of a natural language query system (NLQS). Subsequently, he founded Phoenix Solutions, where he guided the development of algorithms for statistics- and semantics-based signal processing of speech that allow a computer to take in natural speech questions and return answers that also sound like natural speech. Dr. Bennett developed various applications for his technology, including interactive conversational systems and interactive guides, intelligent tutoring systems and form-filling systems. Dr. Bennett is currently at the National Science Foundation serving as a Program Director within the Directorate of Engineering, Division of Industrial Innovation & Partnerships.

7. Defendant Wells Fargo is a financial services company that provides banking, insurance, investment, mortgage loan, and consumer finance services. In connection with its electronic services, Defendant (and/or others on its behalf) established and operates a number of customer support lines, which can be reached for example at (800) 642-4720 and upon information and belief, other toll-free phone numbers. The customer support lines employ a natural language interactive voice response (IVR) system that includes a virtual agent (hereinafter interchangeably referred to as "IVR system").

8. The Plaintiff's natural language IVR system is superior to conventional touch-tone systems because the caller can simply talk to the system using natural language. In contrast, touch-tone IVR systems require the caller to select from a series of choices using a more limited telephone keypad. IVR touch

tone systems are also less efficient since they require callers to listen to an entire menu of choices and wade through a series of menus before providing a response to the caller. Consumers hang up at a greater rate in frustration when they become lost in the maze of menus.

9. The alternative to touch tone menu systems is to employ live operators. When compared to live operators, the Plaintiff's IVR system is much more cost effective. Based upon industry data, it is estimated that Defendant's use of its current IVR system has allowed it to save 93% of the cost it previously incurred in providing its customer support line and Defendant's customer satisfaction has increased by 30%.

10. Upon information and belief, Defendant operates its IVR system using a combination of telephony hardware and computer server hardware that is specifically adapted by Defendant (and/or others on its behalf) to respond to spoken questions from callers concerning the Defendant's business. Such hardware uses supporting software that includes speech recognition and natural language engines used to understand the spoken questions from callers.

11. Upon information and belief, the speech recognition engine used by Defendant is distributed, so that some of the speech-processing operations for understanding callers are performed on a client computing system (such as telephony platform or other hardware) while other speech processing operations are performed on a separate server computing system. Upon information and belief, Defendant (and/or others on its behalf) configure such computing systems to customize what speech processing operations will take place on such respective hardware systems to maximize certain characteristics of the system, and to regulate how speech data from the callers is transferred between such systems.

12. When customers place calls to Defendant's IVR system, they can speak in a conversational style as if they were speaking to a real person.

Defendant's interactive virtual agent responds to the caller's questions in real-time by providing answers in natural speech. The virtual agent has been taught natural language dialogues based on information concerning Defendant's products provided by the Defendant and incorporated into the software. In this manner, the virtual agent can understand questions posed by customers concerning Defendant's products, and give relevant answers.

13. Defendant's IVR system uses a speech recognition engine to break down the customer's questions into specific words understood by the IVR system. For example, the speech recognition engine could determine that the user has said his or her account number. Defendant controls precisely what specific words its IVR system will understand as part of its vocabulary by configuring (and/or having others configure on its behalf) certain aspects of such client computing system and/or server computing system.

14. Defendant's IVR system employs a natural language engine to understand the meaning of the specific words spoken by its customers. The IVR system, by understanding the meaning and context of specific words, may determine that the customer is asking about a service related problem. Defendant controls precisely what interpretation the IVR system should give to various words spoken by its customers by configuring (and/or having others configure on its behalf) certain aspects of the client computing system and/or server computing system.

15. Based on determining the most likely meaning of the customer's specific question, the interactive virtual agent responds with a specific answer. The answer may take the form of an audible response from the agent, or it may take the form of the IVR system routing the caller to a live person working within the appropriate department (such as the service department in the example above). In all instances, Defendant alone controls precisely what responses and actions virtual

agent takes, and has configured (and/or has had others configure on its behalf) certain aspects of such client computing system and/or server computing system to provide such desired responses or actions.

16. Upon information and belief, Defendant also configured and controlled (and/or has had others configure and/or control on its behalf) other aspects of the virtual agent's overall behavior, including among other things, the gender, apparent age, speech rate, prosody, style and rate of response. These parameters are selected and controlled by Defendant to increase customer satisfaction with the customer support line.

17. Upon information and belief, Defendant (and/or others on its behalf) designed, customized and selected the personality exhibited by the virtual agent as well. This electronic persona was specifically selected to be appealing and attractive to Defendant's customers and to maximize utilization of the IVR system by such customers.

18. Upon information and belief, the information used by Defendant's IVR system (including e.g., the grammar used, specific questions to which it can respond, the interpretation of questions, and the answers to be given to customers) were derived by Defendant (and/or others on its behalf) from collecting and studying data from thousands of actual calls made to Defendant's customer support line. Based on this, Plaintiff believes that Defendant (and/or others on its behalf) has trained the IVR system with Defendant's call center data that is unique to Defendant's business. As a result, the IVR system is tailored to respond with appropriate answers to questions posed by Defendant's customer base.

19. Accordingly, Defendant's IVR system has been customized with customer content data that is not available from a third party. This Defendant-specific content data is critically important to the behavior and operation of Defendant's IVR system, since without it the IVR system would not know what

words to recognize from a caller's utterance, how to determine the meaning of such words, and/or what answer to give to the caller as a response.

20. Defendant's IVR system, as noted above, is a combination of components, including at least some hardware, software and content which it obtained from third parties (third party components). Nonetheless, and on information and belief, Defendant is responsible for and has caused such third party components to be combined, adapted and configured (including with such Defendant-specific content) in accordance with specific performance, content requirements and scenarios of the Defendant's customer support operations.

21. Consequently, and on further information and belief, the current structure and operation of Defendant's IVR system is a result of content contributions, performance specifications and operational specifications provided by Defendant and configuration/modification of third party components made by Defendant (and/or others on its behalf). Such third party components – as currently available from such third parties - by themselves would not be sufficient to implement Defendant's IVR system without Defendant's cooperation, contributions and actions, including Defendant's provision of the Defendant-specific content data.

22. On or about June 2, 2006, Plaintiff sent a letter to Defendant, stating that the IVR system is covered by one or more claims of the Patents in Suit. In that letter, Plaintiff included a number of supporting materials to explain its position on the Patents, and further extended an offer to license the Patents in Suit to Defendant. On or about June 27, 2006, Defendant responded, informing Plaintiff that it needed to investigate the matter and requested identification of the patent claims that may be infringed. On or about June 29, 2006, Plaintiff responded to Defendant, stating that Defendant may have overlooked the CD enclosed with the original letter which has extensive representative claim charts pointing out

particularly which claims Plaintiff believes are pertinent to Defendant's system and why. Some many months later on October 18, 2007, and having not heard from Defendant, Plaintiff sent another letter to Defendant to again negotiate a license and requested a response by no later than December 14, 2007. Defendant failed to respond in any meaningful way to the licensing offer or the charge of infringement, necessitating the filing of this action.

III. JURISDICTION AND VENUE

23. This Court has original subject matter jurisdiction over Plaintiff's patent infringement claim pursuant to 28 U.S.C. §1338(a).

24. This Court has personal jurisdiction over Defendant because Defendant's corporate headquarters are located in San Francisco, CA.

25. Venue properly lies in the Northern District of California pursuant to 28 U.S.C. §1391 and §1400, because the acts complained of herein have been committed and are being committed in this Judicial District and Defendant is subject to personal jurisdiction within the District.

IV. FIRST COUNT FOR INFRINGEMENT OF UNITED STATES PATENT NO. 6,633,846

26. Plaintiff hereby incorporates by reference the allegations contained in paragraphs 1 through 25.

27. Plaintiff is the assignee of the U.S. Patent No. 6,633,846 ("the '846 Patent"), attached hereto as Exhibit 1, entitled "Distributed Real Time Speech Recognition System". Plaintiff owns and has standing and capacity to sue and recover damages for infringement under the '846 Patent.

28. Defendant has violated Plaintiff's patent rights by operating an IVR system covered by at least one claim of the '846 Patent. Wells Fargo's infringing IVR system has not been manufactured or authorized in any manner by the Plaintiff.

29. As a legal consequence of Defendant's infringement, Plaintiff is entitled to compensation for no less than a reasonable royalty, as well as pre-judgment interest and a preliminary and permanent injunction. In the event that the Court does not exercise its equitable discretion to award a permanent injunction, then Plaintiff is entitled to a judgment that includes a sum equal to the total projected value of a compulsory license for the life of the patent at a royalty rate to be determined by a jury, discounted to present value, to compensate Plaintiff for future infringement.

30. The infringement of the '846 Patent has been willful in that Defendant is fully aware of Plaintiff's rights, yet has continued to use the infringing IVR system in violation of the patent laws without a good faith basis for believing it does not infringe or the patent is invalid. This intentional refusal to respect Plaintiff's patent rights constitutes willful infringement under 35 U.S.C. §§ 284 and 285, thereby entitling Plaintiff to treble damages and attorneys' fees.

V. SECOND COUNT FOR INFRINGEMENT OF UNITED STATES PATENT NO. 6,665,640

31. Plaintiff hereby incorporates by reference the allegations contained in paragraphs 1 through 25.

32. Plaintiff is the assignee of the U.S. Patent No. 6,665,640 ("the '640 Patent"), attached hereto as Exhibit 2, entitled "Interactive Speech Based Learning/Training System Formulating Search Queries Based on Natural Language Parsing of Recognized User Queries". Plaintiff owns and has standing and capacity to sue and recover damages for infringement under the '640 Patent.

33. Defendant has violated Plaintiff's patent rights by operating an IVR system covered by at least one claim of the '640 Patent. Wells Fargo's infringing IVR system has not been manufactured or authorized in any manner by the Plaintiff.

34. As a legal consequence of Defendant's infringement, Plaintiff is entitled to compensation for no less than a reasonable royalty, as well as pre-judgment interest and a preliminary and permanent injunction. In the event that the Court does not exercise its equitable discretion to award a permanent injunction, then Plaintiff is entitled to a judgment that includes a sum equal to the total projected value of a compulsory license for the life of the patent at a royalty rate to be determined by a jury, discounted to present value, to compensate Plaintiff for future infringement.

35. The infringement of the '640 Patent has been willful in that Defendant is fully aware of Plaintiff's rights, yet has continued to use the infringing IVR system in violation of the patent laws without a good faith basis for believing it does not infringe or the patent is invalid. This intentional refusal to respect Plaintiff's patent rights constitutes willful infringement under 35 U.S.C. §§ 284 and 285, thereby entitling Plaintiff to treble damages and attorneys' fees.

VI. THIRD COUNT FOR INFRINGEMENT OF UNITED STATES PATENT NO. 7,050,977

36. Plaintiff hereby incorporates by reference the allegations contained in paragraphs 1 through 25.

37. Plaintiff is the assignee of the U.S. Patent No. 7,050,977 ("the '977 Patent"), attached hereto as Exhibit 3, entitled "Speech-Enabled Server for Internet Website and Method". Plaintiff owns and has standing and capacity to sue and recover damages for infringement under the '977 Patent.

38. Defendant has violated Plaintiff's patent rights by operating an IVR system covered by at least one claim of the '977 Patent. Wells Fargo's infringing IVR system has not been manufactured or authorized in any manner by the Plaintiff.

39. As a legal consequence of Defendant's infringement, Plaintiff is entitled to compensation for no less than a reasonable royalty, as well as pre-judgment interest and a preliminary and permanent injunction. In the event that the Court does not exercise its equitable discretion to award a permanent injunction, then Plaintiff is entitled to a judgment that includes a sum equal to the total projected value of a compulsory license for the life of the patent at a royalty rate to be determined by a jury, discounted to present value, to compensate Plaintiff for future infringement.

40. The infringement of the '977 Patent has been willful in that Defendant is fully aware of Plaintiff's rights, yet has continued to use the infringing IVR system in violation of the patent laws without a good faith basis for believing it does not infringe or the patent is invalid. This intentional refusal to respect Plaintiff's patent rights constitutes willful infringement under 35 U.S.C. §§ 284 and 285, thereby entitling Plaintiff to treble damages and attorneys' fees.

VII. FOURTH COUNT FOR INFRINGEMENT OF UNITED STATES PATENT NO. 7,277,854

41. Plaintiff hereby incorporates by reference the allegations contained in paragraphs 1 through 25.

42. Plaintiff is the assignee of the U.S. Patent No. 7,277,854 ("the '854 Patent"), attached hereto as Exhibit 4, entitled "Speech Recognition System Interactive Agent". Plaintiff owns and has standing and capacity to sue and recover damages for infringement under the '854 Patent.

43. Defendant has violated Plaintiff's patent rights by operating an IVR system covered by at least one claim of the '854 Patent. Wells Fargo's infringing IVR system has not been manufactured or authorized in any manner by the Plaintiff.

44. As a legal consequence of Defendant's infringement, Plaintiff is entitled to compensation for no less than a reasonable royalty, as well as pre-judgment interest and a preliminary and permanent injunction. In the event that the Court does not exercise its equitable discretion to award a permanent injunction, then Plaintiff is entitled to a judgment that includes a sum equal to the total projected value of a compulsory license for the life of the patent at a royalty rate to be determined by a jury, discounted to present value, to compensate Plaintiff for future infringement.

45. The infringement of the '854 Patent has been willful in that Defendant is fully aware of Plaintiff's rights, yet has continued to use the infringing IVR system in violation of the patent laws without a good faith basis for believing it does not infringe or the patent is invalid. This intentional refusal to respect Plaintiff's patent rights constitutes willful infringement under 35 U.S.C. §§ 284 and 285, thereby entitling Plaintiff to treble damages and attorneys' fees.

VIII. DEMAND FOR JURY TRIAL

46. Plaintiff hereby exercises its right to a jury trial under the Seventh Amendment to the United States Constitution, and pursuant to Fed. R. Civ. Proc., Rule 38, demands a jury trial in accordance therewith.

IX. PRAYER FOR RELIEF

WHEREFORE, Plaintiff prays for:

a. A preliminary injunction, barring Defendant and all of its agents, officers, attorneys, successors, and assigns from manufacturing, importing or using any system (or components thereof) that infringes upon the '846, the '640, the '977 and the '854 Patents;

b. A permanent injunction, barring Defendant and all of its agents, officers, successors and assigns from manufacturing, importing or using any system

(or components thereof) that infringes upon the '846, the '640, the '977 and the '854 Patents;

c. That Defendant be required to account to Plaintiff for all savings and revenues realized by Defendant and any subsidiary and any partner company of Defendant from the use of IVR systems infringing the '846, the '640, the '977 and the '854 Patents;

d. A judgment for compensatory damages, not less than reasonable royalty, suffered as a result of the patent infringement as well as prejudgment interest;

e. A judgment including a sum equal to a the total projected value of a compulsory license for the life of the patents, discounted to present value, to compensate Plaintiff for future infringement in the event that a permanent injunction is not awarded;

f. Treble damages and attorneys' fees pursuant to 35 U.S.C. §§ 284 and 285 for willful infringement of the '846, the '640, the '977 and the '854 Patents by Defendant; and,

g. Any and all other relief that the Court deems proper.

Respectfully submitted,

TROJAN LAW OFFICES

by

Dated: February 6, 2008

R. Joseph Trojan
Attorney for Plaintiff,
PHOENIX SOLUTIONS, INC.

EXHIBIT 1

US006633846B1

(12) **United States Patent**
Bennett et al.

(10) **Patent No.: US 6,633,846 B1**
(45) **Date of Patent: Oct. 14, 2003**

(54) **DISTRIBUTED REALTIME SPEECH RECOGNITION SYSTEM**

(75) Inventors: **Ian M. Bennett**, Palo Alto, CA (US); **Bandi Ramesh Babu**, Anantapur (IN); **Kishor Morkhandikar**, Gulbarga (IN); **Pallaki Gururaj**, Bangalore (IN)

(73) Assignee: **Phoenix Solutions, Inc.**, Palo Alto, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/439,145**

(22) Filed: **Nov. 12, 1999**

(51) **Int. Cl.**[7] **G10L 15/02**; G10L 15/18
(52) **U.S. Cl.** **704/257**; 704/270.1
(58) **Field of Search** 704/9, 10, 232, 704/251, 255, 256, 257, 270, 275, 270.1; 434/185

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,473,904 A 9/1984 Suehiro et al.

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

EP	1 094 388 A2	4/2001
EP	1 096 471 A1	5/2001
WO	9811534	3/1998
WO	WO 99/48011 A1	9/1999
WO	WO 99/50830	10/1999
WO	WO 00/14727 A1	3/2000
WO	WO 00/17854 A1	3/2000
WO	WO 00/20962 A2	4/2000
WO	WO 00/21075 A1	4/2000
WO	WO 00/21232 A2	4/2000
WO	WO 00/22610 A1	4/2000
WO	WO 00/30072 A2	5/2000
WO	WO 00/30287 A1	5/2000
WO	WO 00/68823 A2	11/2000
WO	WO 01/16936 A1	3/2001
WO	WO 01/18693 A2	3/2001
WO	WO 01/26093 A1	4/2001
WO	WO 01/78065 A1	10/2001
WO	WO 01/95312 A1	12/2001
WO	WO 02/03380 A1	1/2002

OTHER PUBLICATIONS

G.D. Forney, "The Viterbi Algorithm," *Proceedings of the IEEE,* vol. 73, pp. 268–278, Mar. 1973.

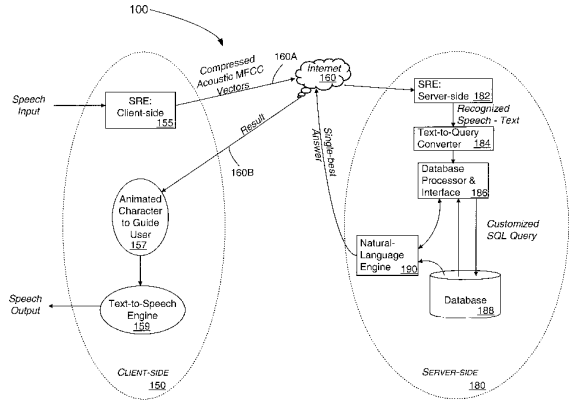
(List continued on next page.)

Primary Examiner—Marsha D. Banks-Harold
Assistant Examiner—Martin Lerner
(74) *Attorney, Agent, or Firm*—J. Nicholas Gross

(57) **ABSTRACT**

A real-time system incorporating speech recognition and linguistic processing for recognizing a spoken query by a user and distributed between client and server, is disclosed. The system accepts user's queries in the form of speech at the client where minimal processing extracts a sufficient number of acoustic speech vectors representing the utterance. These vectors are sent via a communications channel to the server where additional acoustic vectors are derived. Using Hidden Markov Models (HMMs), and appropriate grammars and dictionaries conditioned by the selections made by the user, the speech representing the user's query is fully decoded into text (or some other suitable form) at the server. This text corresponding to the user's query is then simultaneously sent to a natural language engine and a database processor where optimized SQL statements are constructed for a full-text search from a database for a recordset of several stored questions that best matches the user's query. Further processing in the natural language engine narrows the search to a single stored question. The answer corresponding to this single stored question is next retrieved from the file path and sent to the client in compressed form. At the client, the answer to the user's query is articulated to the user using a text-to-speech engine in his or her native natural language. The system requires no training and can operate in several natural languages.

57 Claims, 31 Drawing Sheets



U.S. PATENT DOCUMENTS

4,587,670	A		5/1986	Levinson et al.	
4,783,803	A		11/1988	Baker et al.	
4,785,408	A		11/1988	Britton et al.	
4,852,170	A		7/1989	Bordeaux	
4,914,590	A		4/1990	Loatman et al.	
4,991,094	A		2/1991	Fagan et al.	
4,991,217	A		2/1991	Garrett et al.	
5,068,789	A		11/1991	Van Vliembergen	
5,146,405	A		9/1992	Church	
5,157,727	A		10/1992	Schloss	
5,231,670	A		7/1993	Goldhor et al.	
5,265,014	A	*	11/1993	Haddock et al.	704/9
5,293,584	A		3/1994	Brown et al.	
5,384,892	A		1/1995	Strong	
5,475,792	A		12/1995	Stanford et al.	
5,513,298	A		4/1996	Stanford et al.	
5,540,589	A	*	7/1996	Waters	704/246
5,602,963	A		2/1997	Bissonnette et al.	
5,680,628	A		10/1997	Carus et al.	
5,727,950	A		3/1998	Cook et al.	
5,737,485	A	*	4/1998	Flanagan et al.	704/232
5,758,322	A		5/1998	Rongley	704/275
5,802,526	A		9/1998	Fawcett et al.	707/104
5,819,220	A		10/1998	Sarukkai et al.	704/243
5,836,771	A		11/1998	Ho et al.	
5,867,817	A		2/1999	Catallo et al.	704/255
5,873,062	A		2/1999	Hansen et al.	
5,884,302	A		3/1999	Ho	
5,915,236	A		6/1999	Gould et al.	704/251
5,934,910	A		8/1999	Ho et al.	
5,956,683	A		9/1999	Jacobs et al.	704/275
5,960,394	A		9/1999	Gould et al.	704/240
5,960,399	A		9/1999	Barclay et al.	704/270
5,995,928	A		11/1999	Nguyen et al.	
6,009,387	A		12/1999	Ramaswamy et al.	
6,029,124	A		2/2000	Gillick et al.	704/200
6,035,275	A		3/2000	Brode et al.	
6,112,176	A		8/2000	Goldenthal et al.	
6,119,087	A		9/2000	Kuhn et al.	
6,138,089	A		10/2000	Guberman	704/207
6,141,640	A		10/2000	Moo	
6,144,848	A		11/2000	Walsh et al.	455/419
6,144,938	A		11/2000	Surace et al.	704/257
6,182,038	B1		1/2001	Balakrishnan et al.	
6,185,535	B1		2/2001	Hedin et al.	
6,233,559	B1		5/2001	Balakrishnan	
6,256,607	B1		7/2001	Digalakis et al.	
6,269,336	B1		7/2001	Ladd et al.	
6,327,561	B1		12/2001	Smith et al.	
6,327,568	B1	*	12/2001	Joost	704/251
6,336,090	B1	*	1/2002	Chou et al.	704/221
6,363,349	B1		3/2002	Urs et al.	
6,374,219	B1		4/2002	Jiang	
6,381,594	B1		4/2002	Eichstaedt et al.	
6,389,389	B1		5/2002	Meunier et al.	
6,408,272	B1		6/2002	White et al.	
6,411,926	B1		6/2002	Chang	
6,427,063	B1		7/2002	Cook et al.	
2001/0016813	A1		8/2001	Brown et al.	
2001/0032083	A1		10/2001	Van Cleven	
2001/0056346	A1		12/2001	Ueyama et al.	
2002/0046023	A1		4/2002	Fujii et al.	
2002/0059068	A1		5/2002	Rose et al.	
2002/0059069	A1		5/2002	Hsu et al.	
2002/0086269	A1		7/2002	Shpiro	
2002/0087325	A1		7/2002	Lee et al.	
2002/0087655	A1		7/2002	Bridgman et al.	
2002/0091527	A1		7/2002	Shiau	

OTHER PUBLICATIONS

J.H. Baker, "The Dragon System—An Overview," *IEEE Trans. on ASSP Processing,* ASSP–23(1): 24–29, Feb. 1975.

I. Bennett, "A Study of Speech Compression Using Analog Time Domain Sampling techniques," *A Dissertation Submitted to the Dept. Of Electrical Engineering and the Committee on Graduate Studies of Stanford University,* May 1975, pp. 16–32; 76–111.

H.R. Rabiner, "Digital Processing of Speech Signals," Prentice Hall, 1978, pp. 116–171; 355–395.

F. Jelinek et al, "Continuous Speech Recognition: Statistical Methods," *Handbook of Statistics,* vol. 2, P.R. Krishnaiah, Ed. Amsterdam, The Netherlands, North–Holland, 1982, pp. 549–573.

L.R. Bahl, F. Jeninek, R.L. Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI–5, pp. 179–190, Mar. 1983.

R.A. Hudson, "Word Grammar," Blackman Inc., Cambridge, MA, 1984, pp. 1–14; 41–42; 76–90; 94–98; 106–109; 211–220.

R. Quirk, S. Greenbaum, G. Leech and J. Svartvik, "A Comprehensive Grammar of English Language," Longman, London and New York, 1985, pp. 245–331.

J. Makhoul, S. Roucos, H. Gish, "Vector Quantization in Speech Coding," *Proceedings of the IEEE,* vol. 73, No. 11, Nov. 1985, pp. 1551–1588.

L. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE,* vol. 77, No. 2, Feb. 1989, pp. 257–286.

A. Gersho and R.M. Gray, "Vector Quantization and Compression," Kluwer Academic Publishers, 1991, pp. 309–340.

H.R. Rabiner and B.H. Juang, "Fundamentals of Speech Recognition," Prentice Hall, 1993, pp. 11–68.

Nelson Morgan, Hervé Bourlard, Steve Renals, Michael Cohen and Horacio Franco, "Hybrid Neural Network/Hidden Markov Model Systems for Continuous Speech Recognition," *Journal of Pattern Recognition and Artificial Intelligence,* vol. 7, No. 4, 1993, pp. 899–916.

Arons, B., "The Design of Audio Servers and Toolkits for Supporting Speech in the User Interface," believed to be published in: Journal of the American Voice I/O Society, pp. 27–41, Mar. 1991.

Hazen, T et al., "Recent Improvements in an Approach to Segment–Based Automatic Language Identification," believed to be published in: Proceedings of the 1994 International Conference on Spoken Language Processing, Yokohama, Japan, pp. 1883–1886, Sep. 1994.

House, D., "Spoken–Language Access to Multimedia (SLAM): A Multimodal Interface to the World–Wide Web," Masters Thesis, Oregon Graduate Institute, Department of Computer Science & Engineering, 59 pages, Apr. 1995.

Julia, L. et al., "HTTP://WWW.SPEECH.SRI.COM/DEMOS/ ATIS.HTML," believed to be published in: Proceedings AAAI'97: Stanford, pp. 72–76, Jul. 1997.

Lau, R. et al, "Webgalaxy–Integrating Spoken Language and Hypertext Navigation," believed to be published in: in Kokkinakis, G. et al., (Eds.) Eurospeech '97, Proceedings of the 5th European Conference on Speech Communication and Technology, Rhodes (Greece), Sep. 22–25, 1997. pp. 883–886, 1997.

Digalakis, V. et al., "Product–Code Vector Quantization of Cepstral Parameters for Speech Recognition over the WWW," believed to be published in: Proc. ICSLP '98, 4 pages, 1998.

Melin, H., "On Word Boundary Detection in Digit–Based Speaker Verification," believed to be published in: Workshop on Speaker Recognition and its Commercial and Forensic Applications (RLA2C), Avignon, France, Apr. 20–23, pp. 46–49, 1998.

Ramaswamy, G. et al., "Compression of Acoustic Features for Speech Recognition in Network Environments," believed to be published in: IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 977–980, Jun. 1998.

Lu, B. et al., "Scalability Issues in the Real Time Protocol (RTP)," Project Report for CPSC 663 (Real Time Systems), Dept. of Computer Science, Texas A & M University, 19 pages, 1999.

Giuliani, D. et al., "Training of HMM with Filtered Speech Material for Hands–Free Recognition," believed to be published in: Proceedings of ICASSP '99, Phoenix, USA, 4 pages, 1999.

Digalakis,V. et al., "Quantization of Cepstral Parameters for Speech Recognition over the World Wide Web," believed to be published in: IEEE Journal on Selected Areas of Communications, 22 pages, 1999.

Tsakalidis, S. et al., "Efficient Speech Recognition Using Subvector Quantization and Discrete–Mixture HMMs," believed to be published in: Proc. ICASSP '99, 4 pages, 1999.

Lin, B. et al., "A Distributed Architecture for Cooperative Spoken Dialogue Agents with Coherent Dialogue State and History," believed to be published in: IEEE Automatic Speech Recognition and Understanding Workshop, Keystone, Colorado, USA, 4 pages, Dec. 1999.

Meunier, J., "RTP Payload Format for Distrubuted Speech Recognition," 48th IETF AVT WG—Aug. 3, 2000, 10 pages, 2000.

Sand Cherry Networks, SoftServer product literature, 2 pages, 2001.

Kim,H. et al., "A Bitstream–Based Front–End for Wireless Speech Recognition on IS–136 Communications System," IEEE Transactions on Speech and Audio Processing, vol. 9, No. 5, pp. 558–568, Jul. 2001.

L. Travis, "Handbook of Speech Pathology," Appleton–Century–Crofts, Inc., 1957, pp. 91–124.

L.E. Baum, T. Petrie, "Statistical Inference for Probabilistic Functions for Finite State Markov Chains," *The Annals of Mathematical Statistics,* 37: 1554–1563, 1966.

P. Lieberman, "Intonation, Perception and Language," Research Monograph No. 38, MIT Press, Cambridge, Mass., 1967, pp. 5–37.

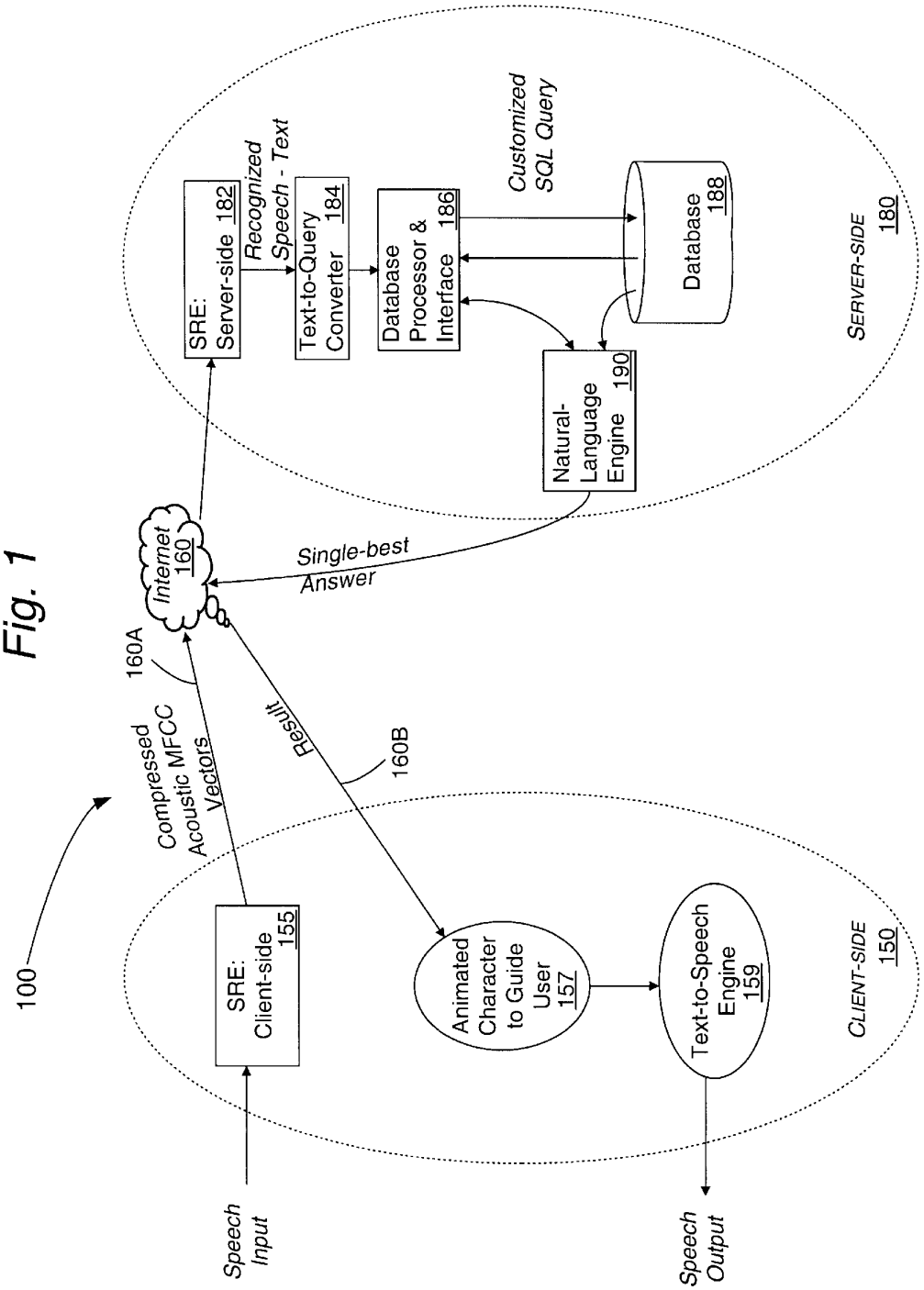
L.E. Baum et al., "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains," *The Annals of Mathematical Statistics,* 1970, vol. 41, No. 1, pp. 164–171.

J.L. Flanagan, "Speech Analysis Synthesis and Perception," 2nd edition, Springer–Verlag Berlin, 1972, pp. 1–53.

L.E. Baum, "An Inequality and Associated Maximation Technique in Statistical Estimation for Probabilistic Functions of Markov Processes," *Inequalities*–III, pp. 1–8, 1972.

* cited by examiner

Fig. 1


100
SRE: Client-side 155
Compressed MFCC Acoustic Vectors
160A
Internet 160
Result
160B
Single-best Answer
SRE: Server-side 182
Recognized Speech - Text
Text-to-Query Converter 184
Database Processor & Interface 186
Customized SQL Query
Database 188
Natural-Language Engine 190
SERVER-SIDE 180
Animated Character to Guide User 157
Text-to-Speech Engine 159
CLIENT-SIDE 150
Speech Input
Speech Output

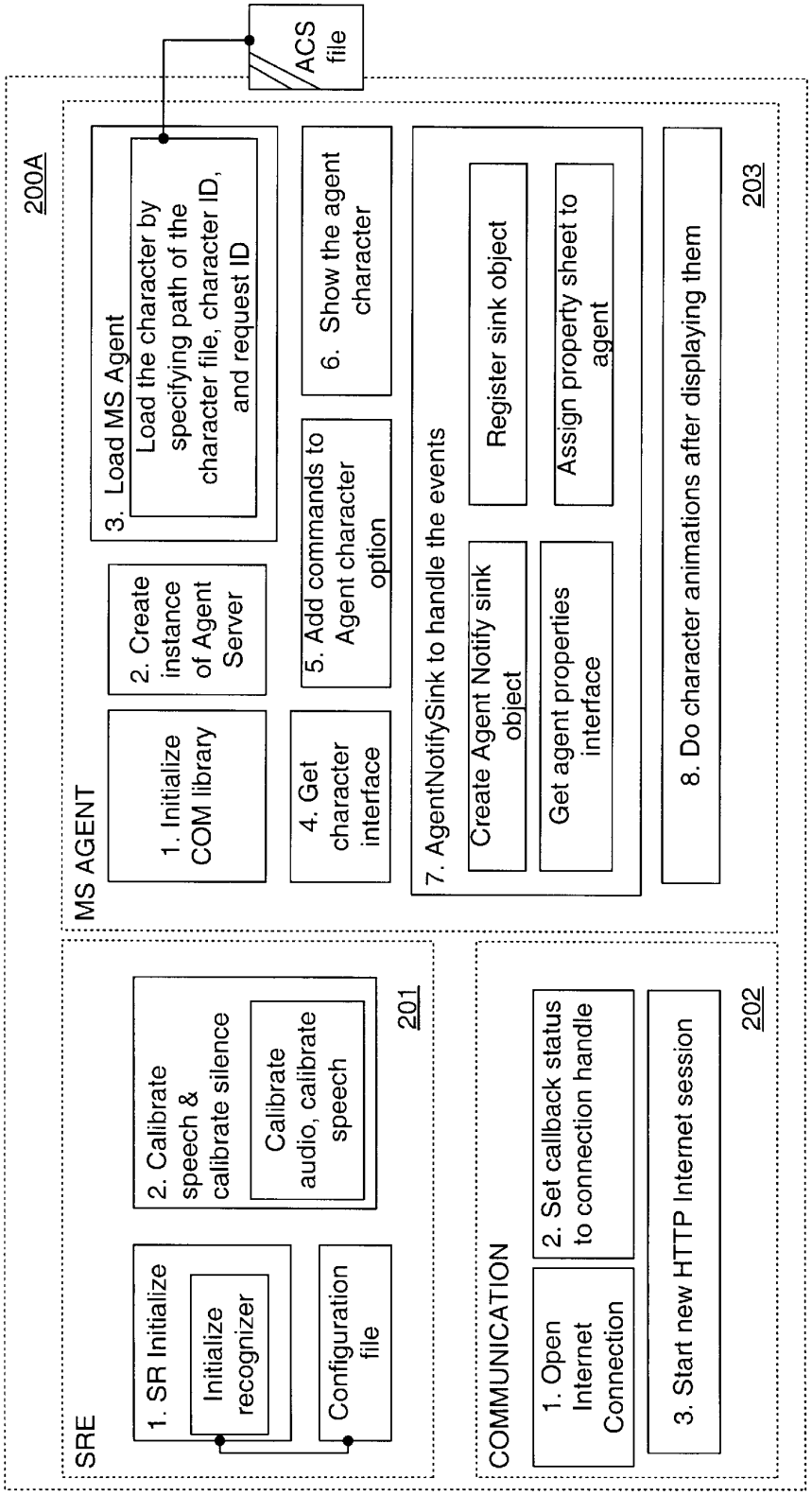


Figure 2A
CLIENT-SIDE SYSTEM LOGIC

Figure 2B

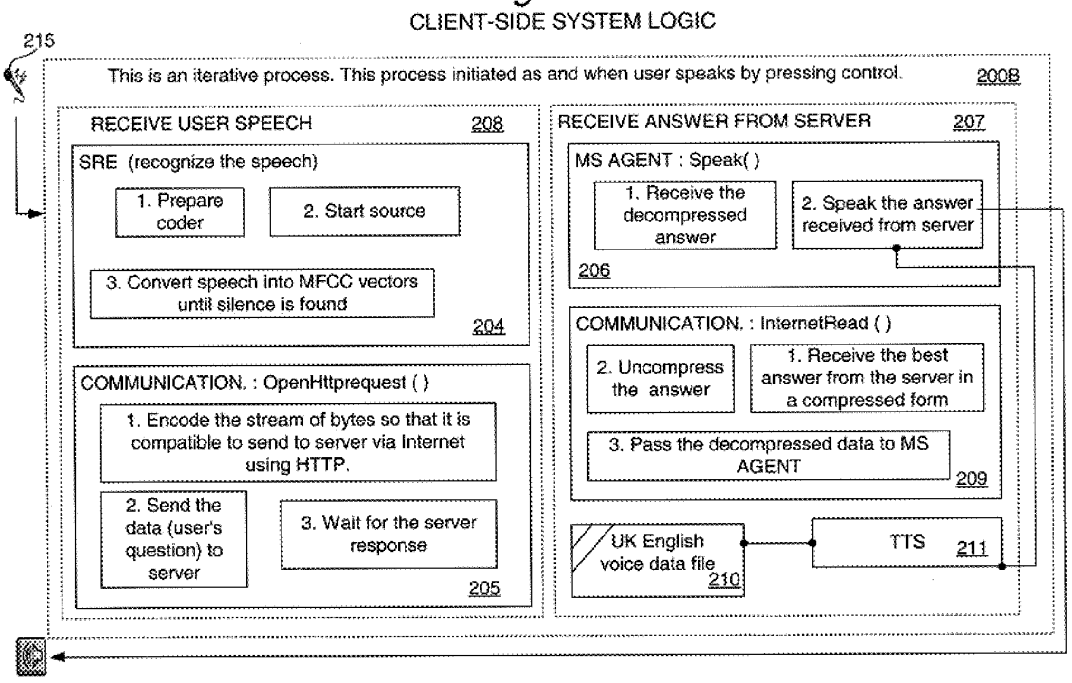

CLIENT-SIDE SYSTEM LOGIC
215
This is an iterative process. This process initiated as and when user speaks by pressing control.
2008
RECEIVE USER SPEECH
208
SRE (recognize the speech)
1. Prepare coder
2. Start source
3. Convert speech into MFCC vectors until silence is found
204
COMMUNICATION. : OpenHttprequest ( )
1. Encode the stream of bytes so that it is compatible to send to server via Internet using HTTP.
2. Send the data (user's question) to server
3. Wait for the server response
205
RECEIVE ANSWER FROM SERVER
207
MS AGENT : Speak( )
1. Receive the decompressed answer
2. Speak the answer received from server
206
COMMUNICATION. : InternetRead ( )
2. Uncompress the answer
1. Receive the best answer from the server in a compressed form
3. Pass the decompressed data to MS AGENT
209
UK English voice data file
210
TTS
211

Figure 2C

CLIENT-SIDE SYSTEM LOGIC

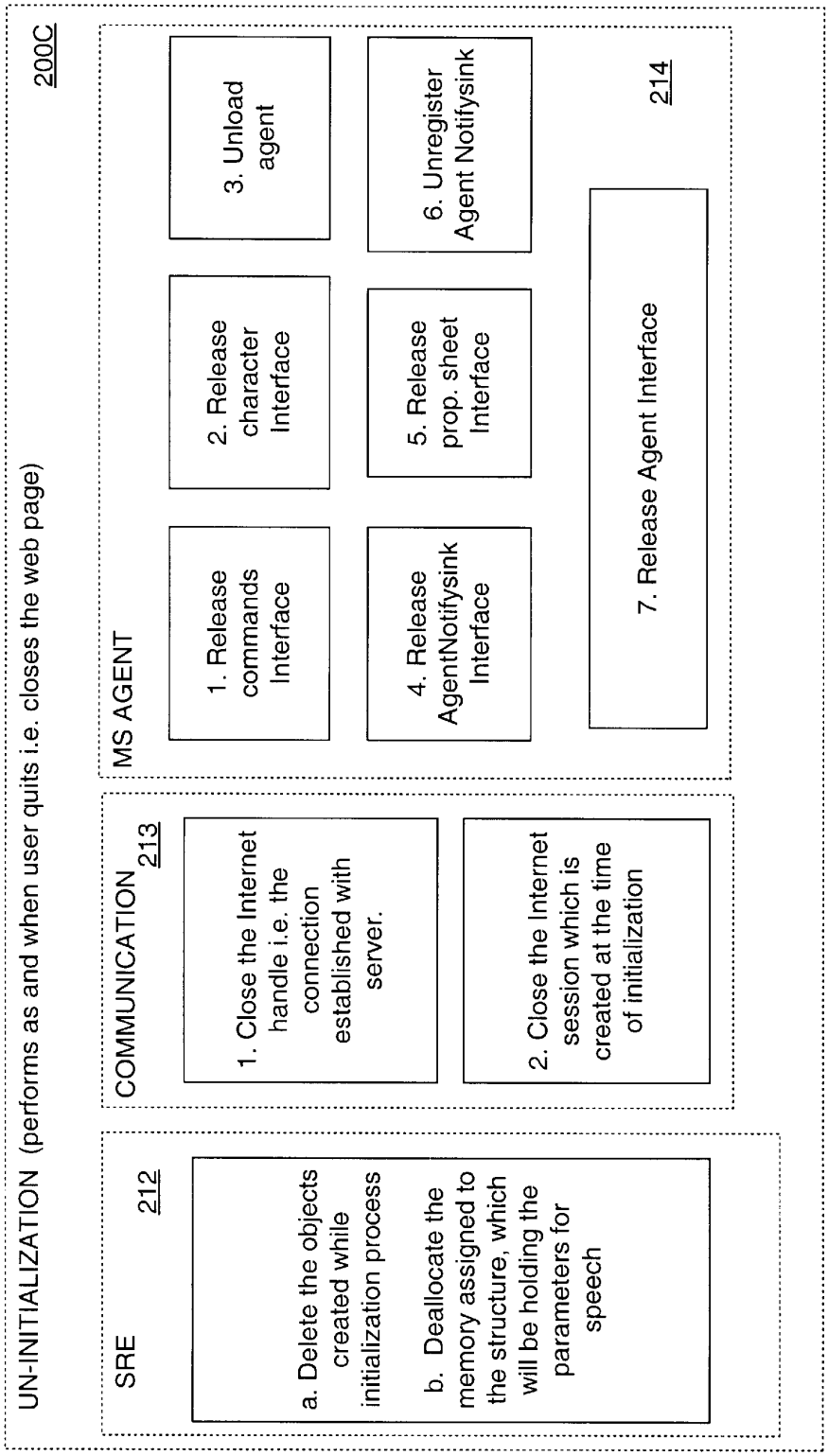


Fig. 2D

Client-side Initialization

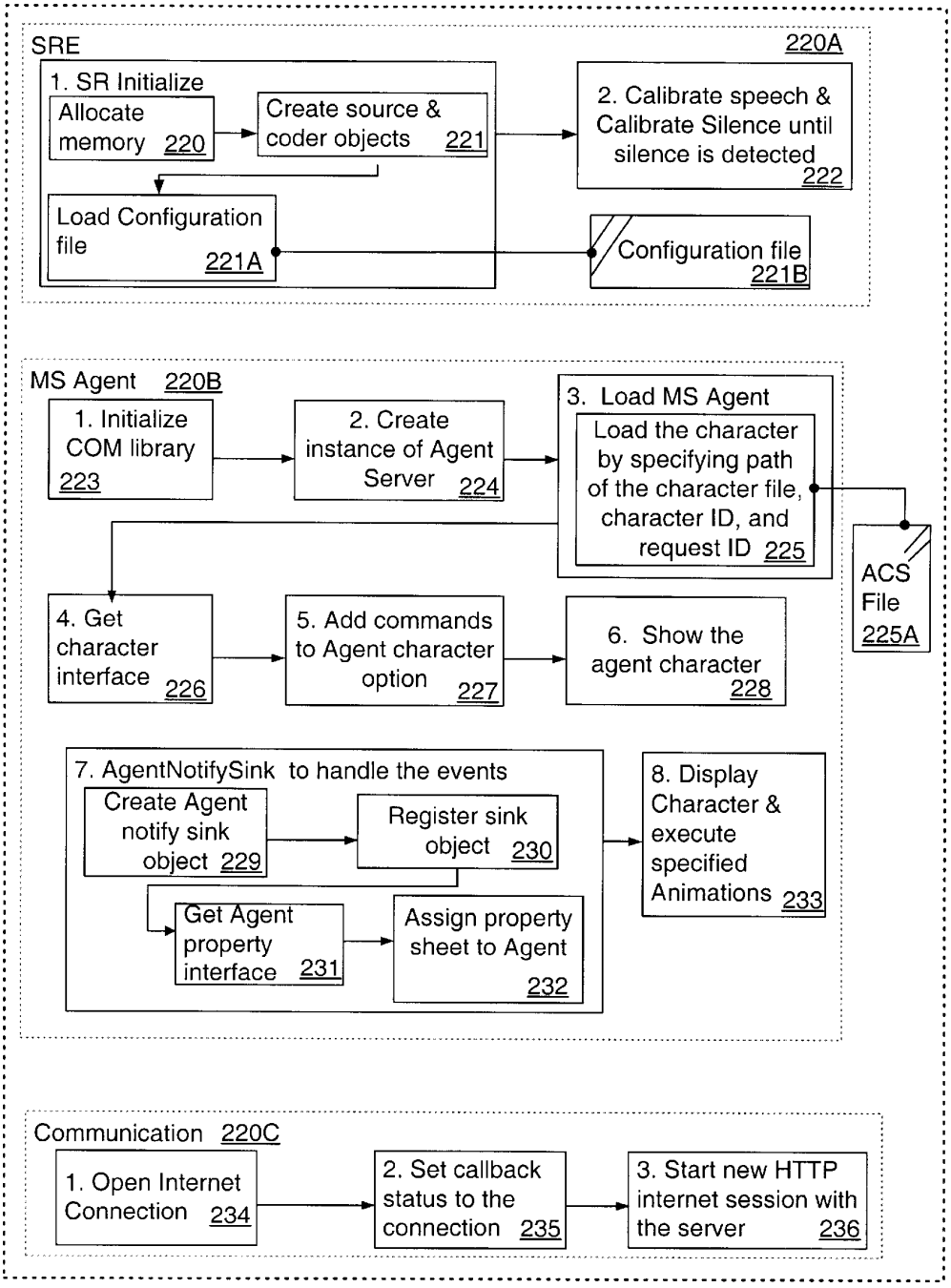


Fig. 3

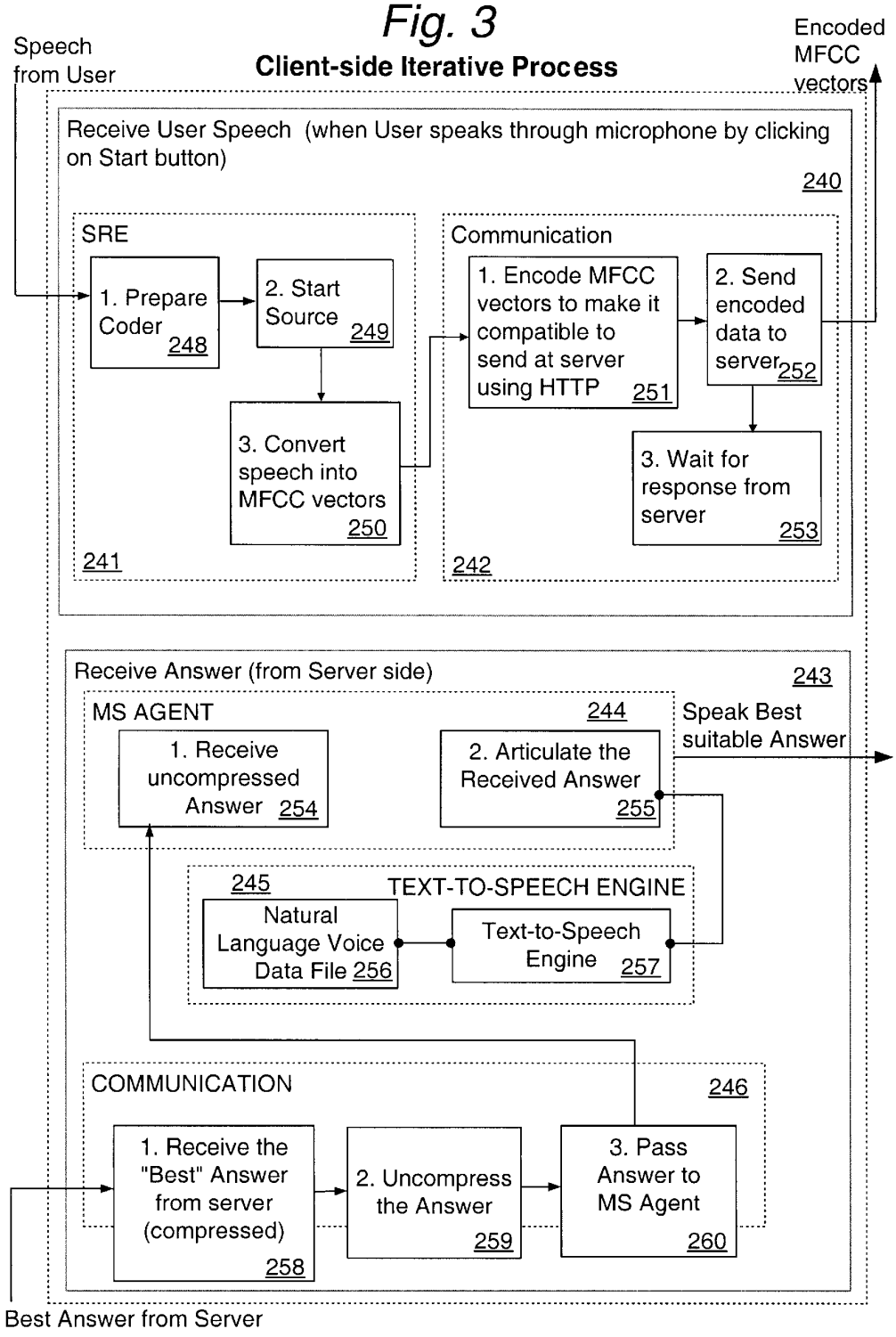

Client-side Iterative Process
Speech from User
Encoded MFCC vectors
Receive User Speech (when User speaks through microphone by clicking on Start button)
240
SRE
1. Prepare Coder 248
2. Start Source 249
3. Convert speech into MFCC vectors 250
241
Communication
1. Encode MFCC vectors to make it compatible to send at server using HTTP 251
2. Send encoded data to server 252
3. Wait for response from server 253
242
Receive Answer (from Server side)
243
MS AGENT
244
1. Receive uncompressed Answer 254
2. Articulate the Received Answer 255
Speak Best suitable Answer
245
TEXT-TO-SPEECH ENGINE
Natural Language Voice Data File 256
Text-to-Speech Engine 257
COMMUNICATION
246
1. Receive the "Best" Answer from server (compressed) 258
2. Uncompress the Answer 259
3. Pass Answer to MS Agent 260
Best Answer from Server

Fig. 4

Client-side Un-Initialization

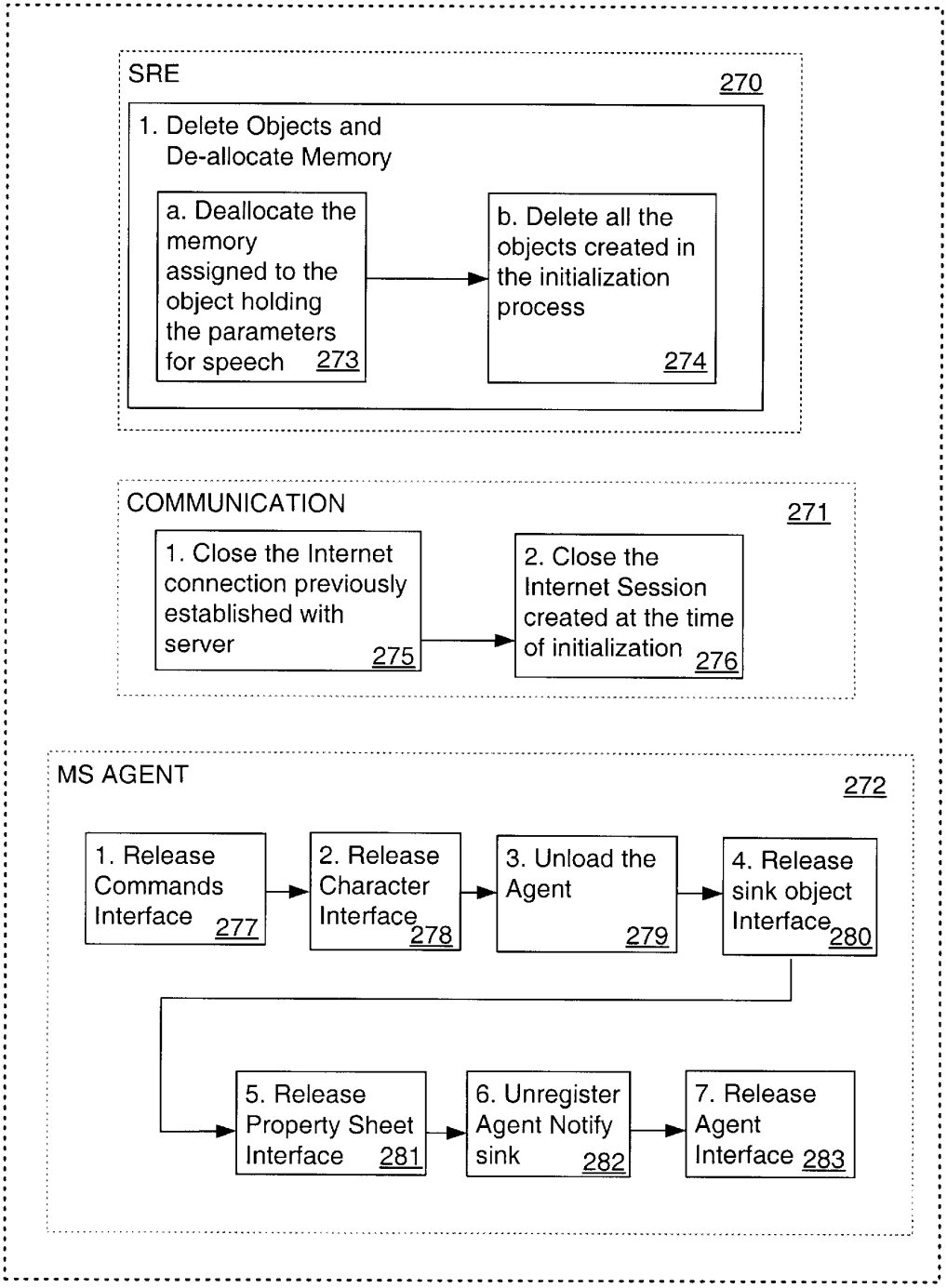
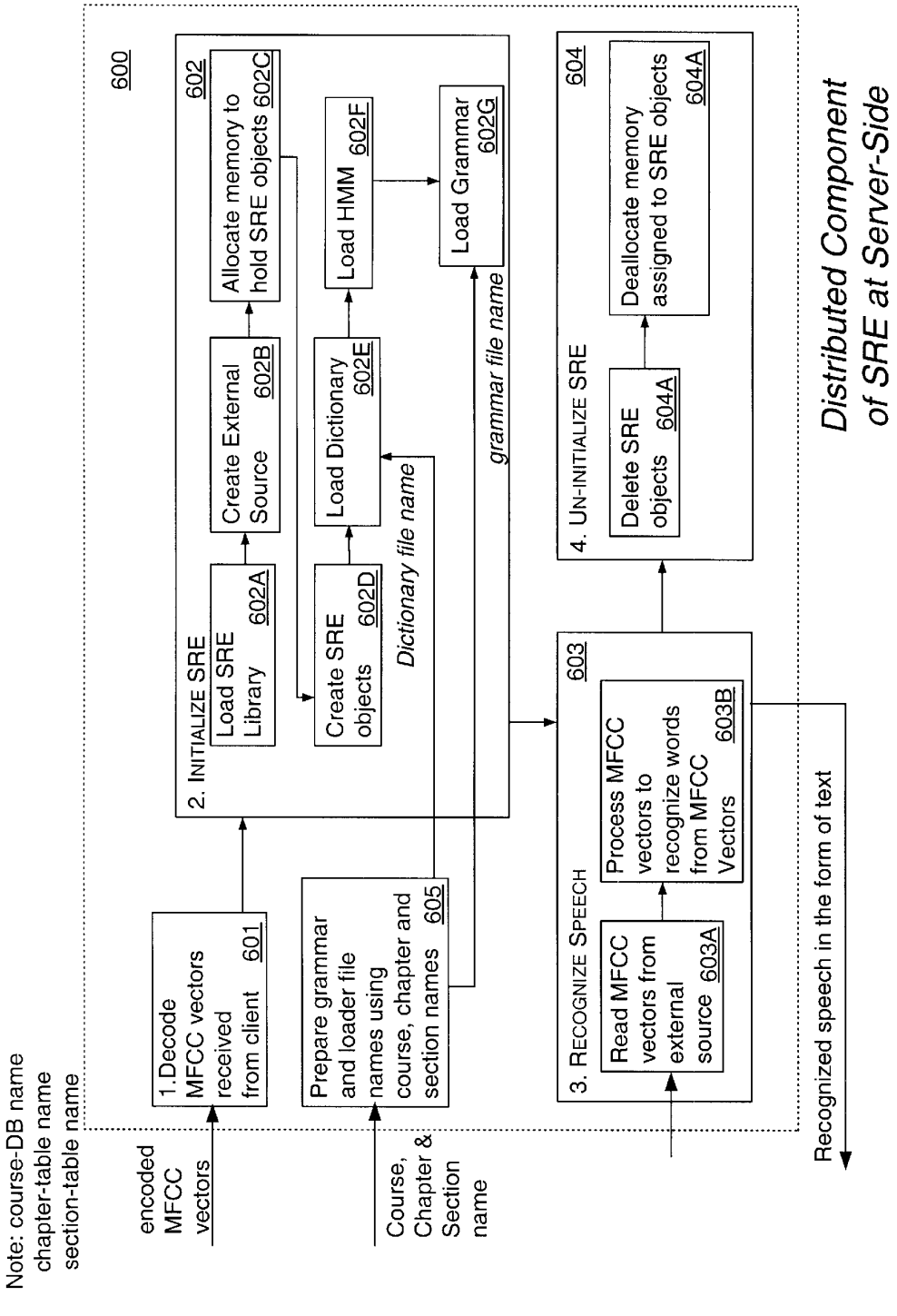


Fig. 4A


600
Distributed Component of SRE at Server-Side
Note: course-DB name
chapter-table name
section-table name
encoded MFCC vectors
Course, Chapter & Section name
Recognized speech in the form of text
1.Decode MFCC vectors received from client 601
2. INITIALIZE SRE 602
Load SRE Library 602A
Create External Source 602B
Allocate memory to hold SRE objects 602C
Create SRE objects 602D
Load Dictionary 602E
Load HMM 602F
Load Grammar 602G
Dictionary file name
grammar file name
Prepare grammar and loader file names using course, chapter and section names 605
3. RECOGNIZE SPEECH 603
Read MFCC vectors from external source 603A
Process MFCC vectors to recognize words from MFCC Vectors 603B
4. UN-INITIALIZE SRE 604
Delete SRE objects 604A
Deallocate memory assigned to SRE objects 604A

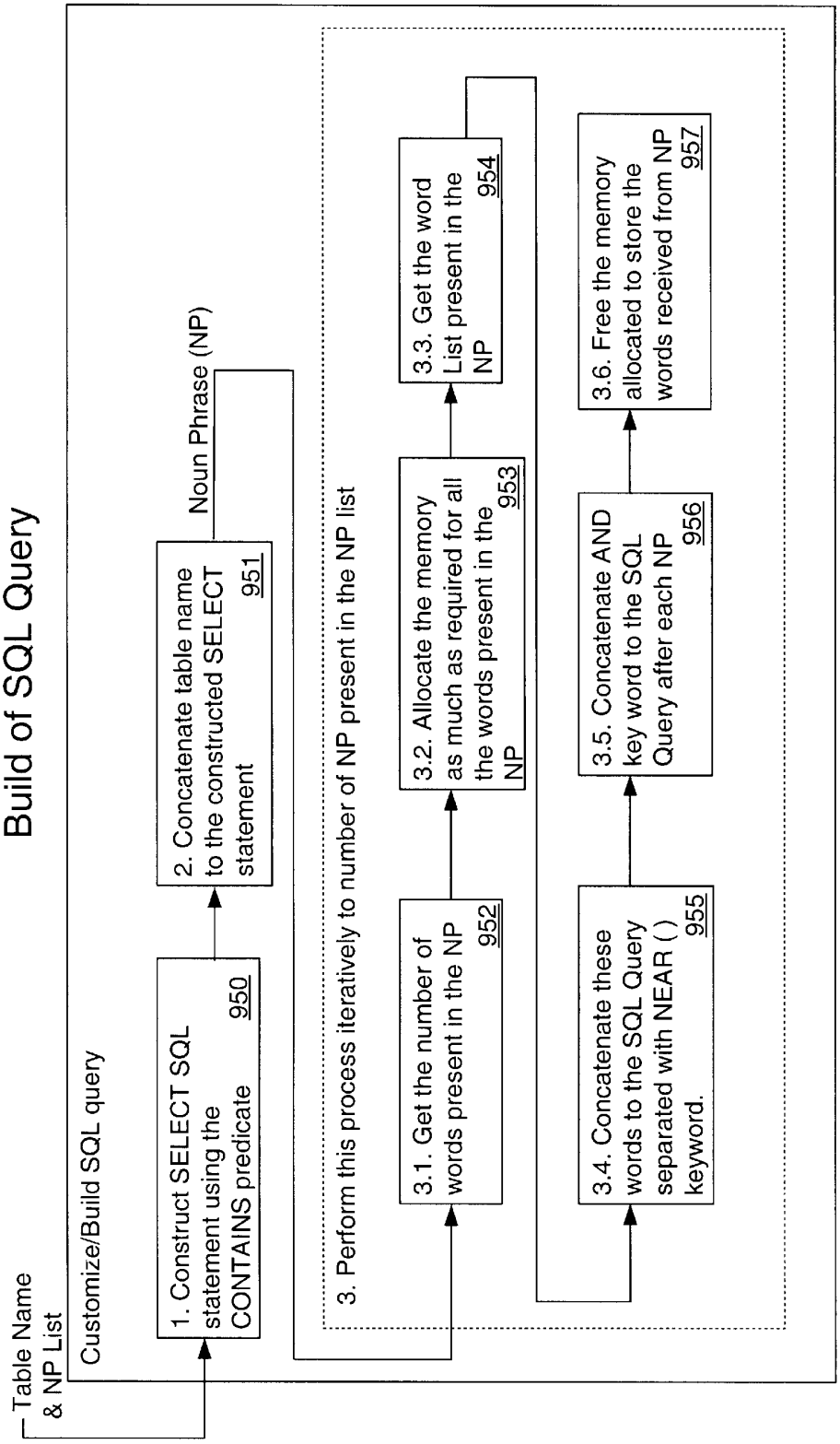


Fig. 4B

Build of SQL Query

Fig. 4C

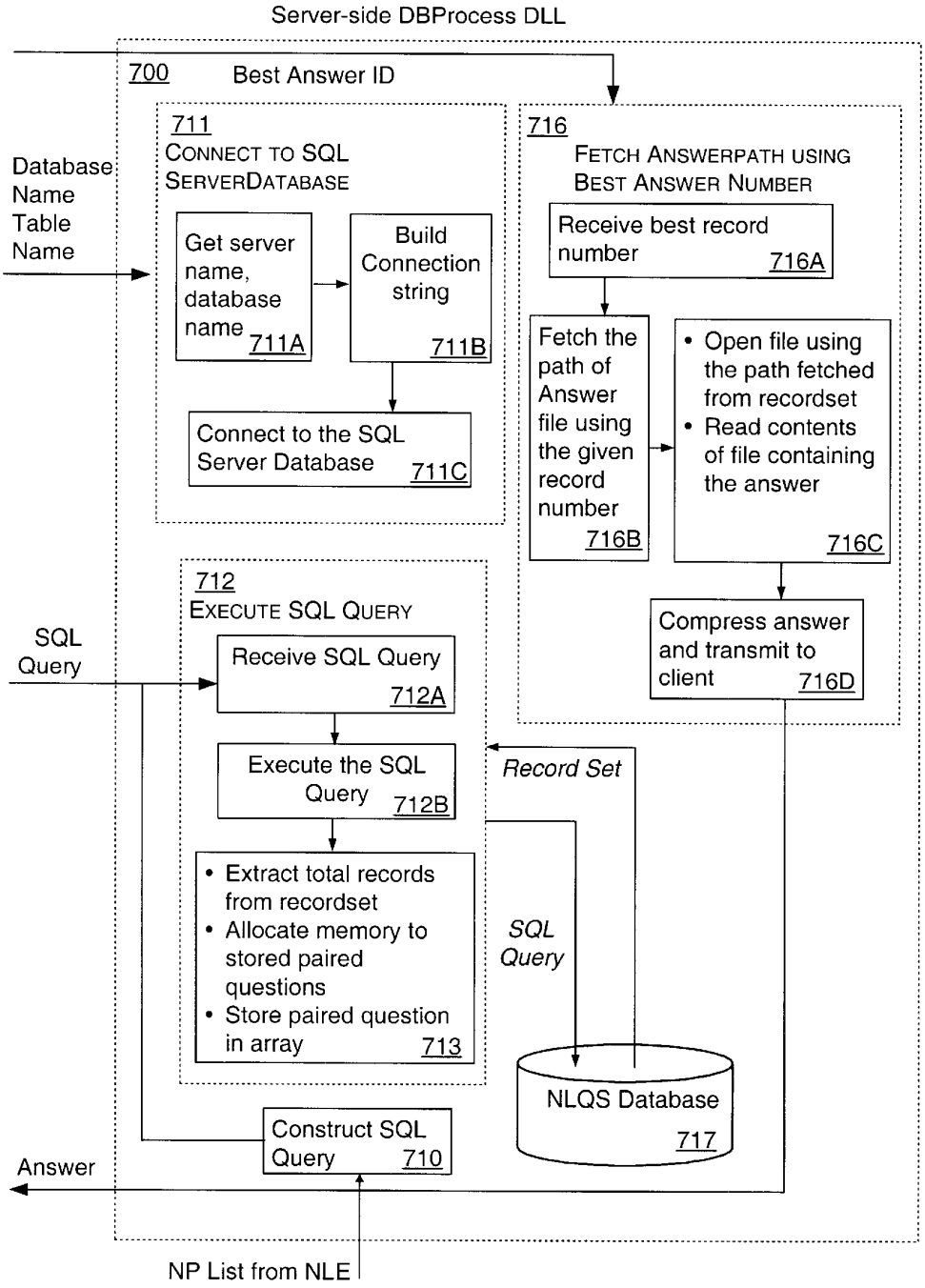

Server-side DBProcess DLL
700
Best Answer ID
711
CONNECT TO SQL SERVERDATABASE
Database Name
Table Name
Get server name, database name 711A
Build Connection string 711B
Connect to the SQL Server Database 711C
716
FETCH ANSWERPATH USING BEST ANSWER NUMBER
Receive best record number 716A
Fetch the path of Answer file using the given record number 716B
• Open file using the path fetched from recordset
• Read contents of file containing the answer 716C
Compress answer and transmit to client 716D
712
EXECUTE SQL QUERY
SQL Query
Receive SQL Query 712A
Execute the SQL Query 712B
Record Set
• Extract total records from recordset
• Allocate memory to stored paired questions
• Store paired question in array 713
SQL Query
NLQS Database 717
Construct SQL Query 710
Answer
NP List from NLE

Fig. 4D

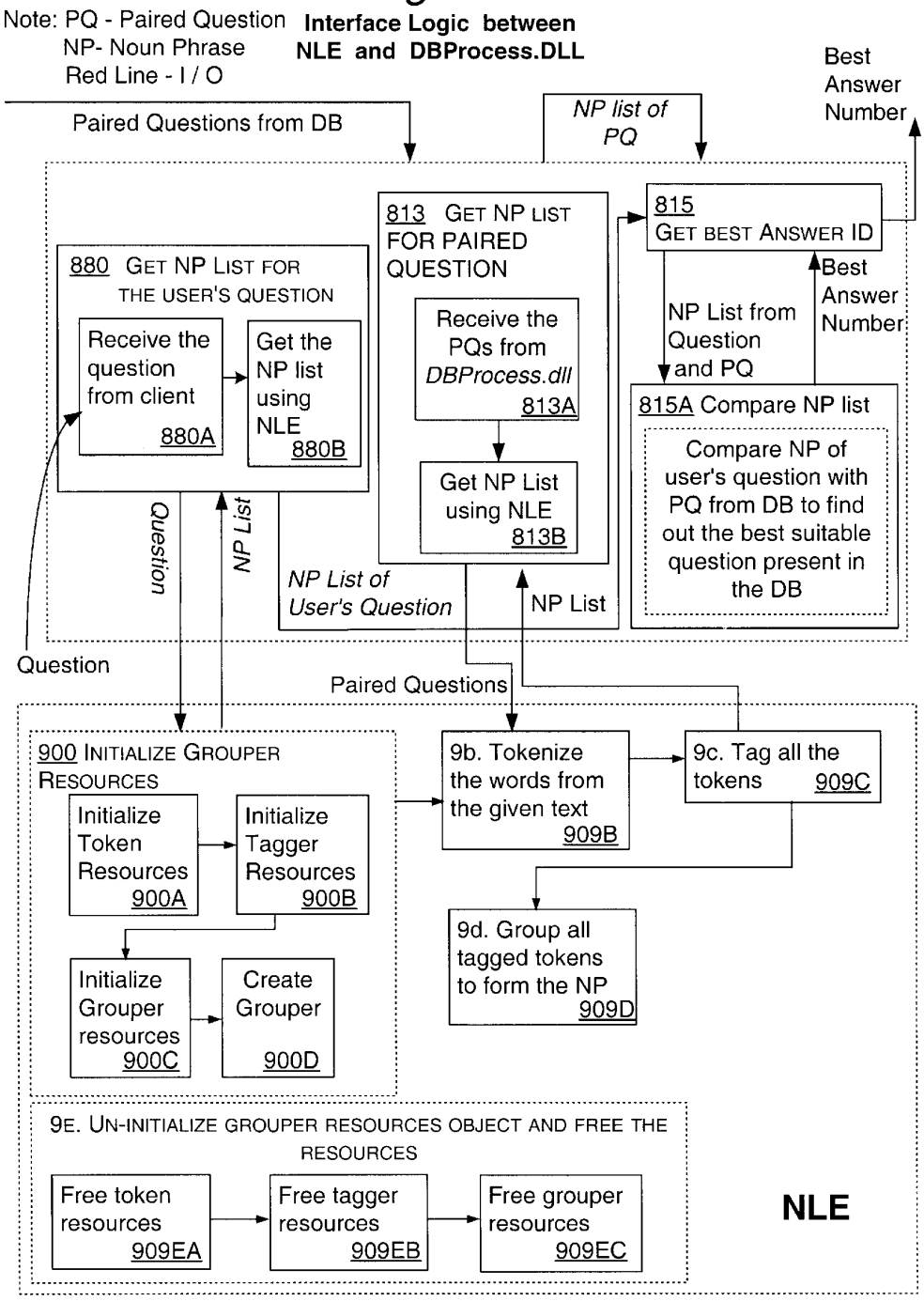

Note: PQ - Paired Question
NP- Noun Phrase
Red Line - I / O
Interface Logic between NLE and DBProcess.DLL
Paired Questions from DB
NP list of PQ
Best Answer Number
880 GET NP LIST FOR THE USER'S QUESTION
Receive the question from client 880A
Get the NP list using NLE 880B
813 GET NP LIST FOR PAIRED QUESTION
Receive the PQs from DBProcess.dll 813A
Get NP List using NLE 813B
815 GET BEST ANSWER ID
NP List from Question and PQ
Best Answer Number
815A Compare NP list
Compare NP of user's question with PQ from DB to find out the best suitable question present in the DB
Question
NP List
NP List of User's Question
NP List
Question
Paired Questions
900 INITIALIZE GROUPER RESOURCES
Initialize Token Resources 900A
Initialize Tagger Resources 900B
Initialize Grouper resources 900C
Create Grouper 900D
9b. Tokenize the words from the given text 909B
9c. Tag all the tokens 909C
9d. Group all tagged tokens to form the NP 909D
9E. UN-INITIALIZE GROUPER RESOURCES OBJECT AND FREE THE RESOURCES
Free token resources 909EA
Free tagger resources 909EB
Free grouper resources 909EC
NLE

Fig. 5

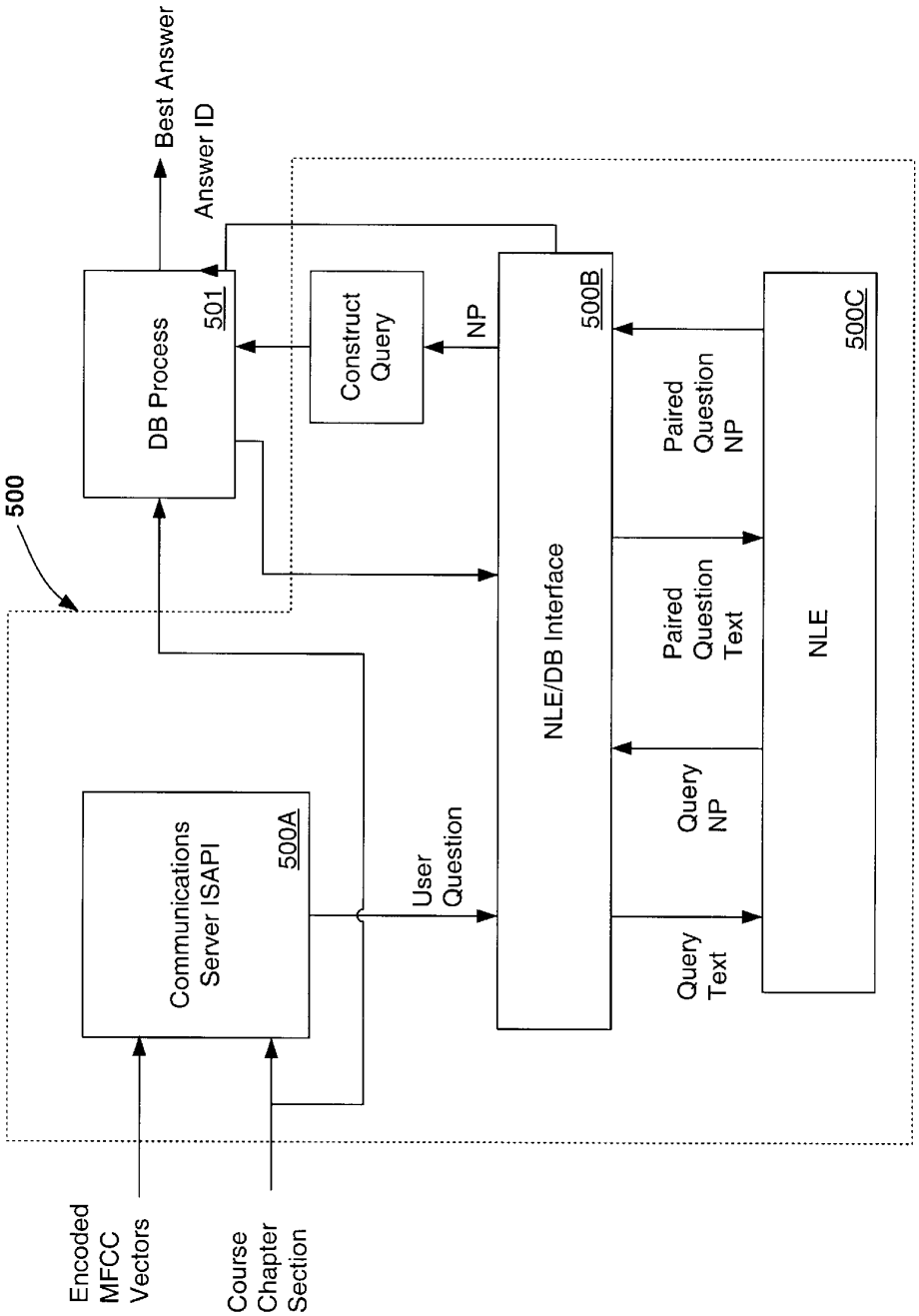

500
Best Answer
Answer ID
DB Process
501
Construct Query
NP
500B
NLE/DB Interface
500C
NLE
Paired Question NP
Paired Question Text
Query NP
Query Text
User Question
Communications Server ISAPI
500A
Encoded MFCC Vectors
Course Chapter Section

Fig.6

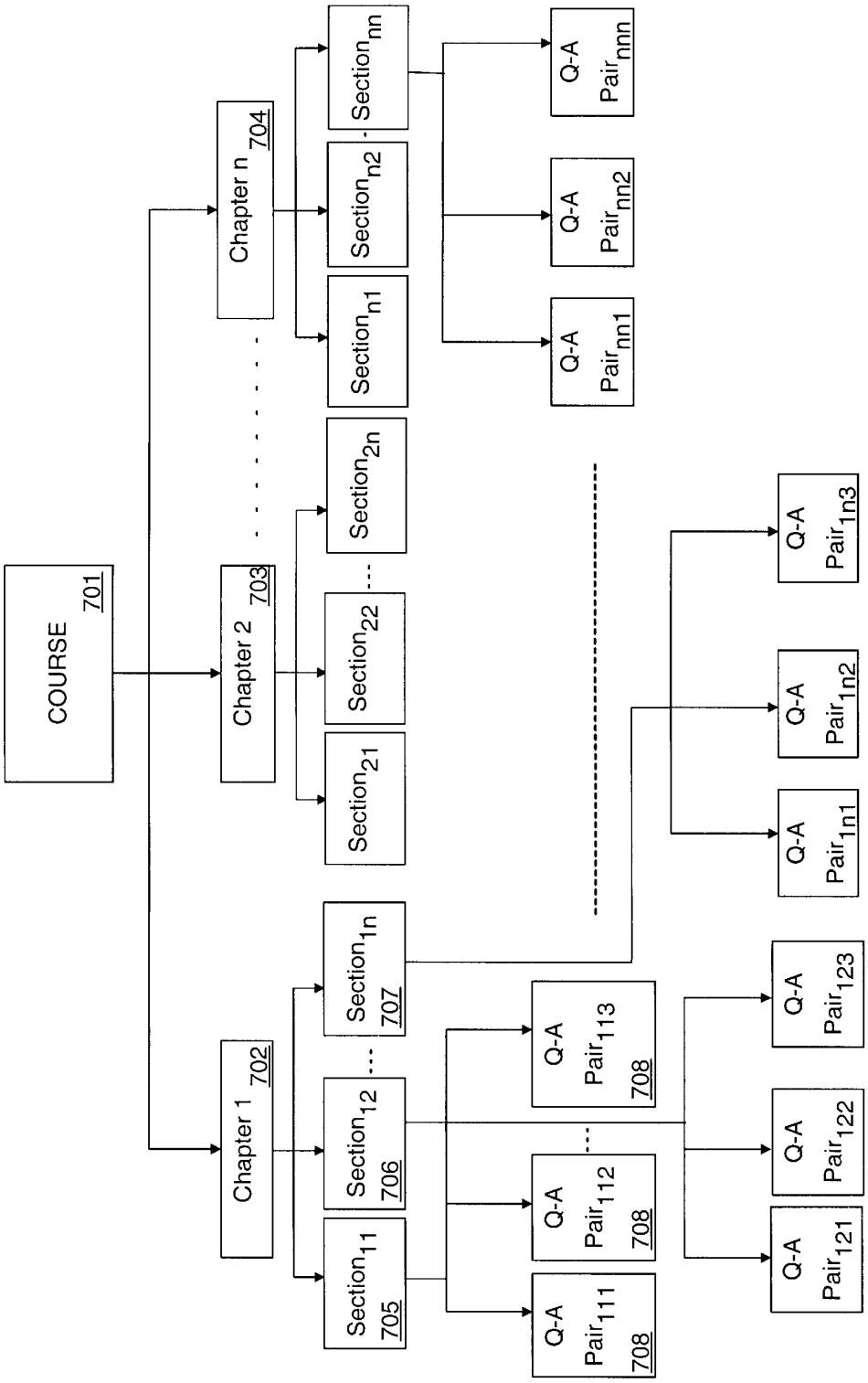

COURSE 701
Chapter 1 702
Chapter 2 703
Chapter n 704
Section11 705
Section12 706
Section1n 707
Section21
Section22
Section2n
Sectionn1
Sectionn2
Sectionnn
Q-A Pair111 708
Q-A Pair112 708
Q-A Pair113 708
Q-A Pair121
Q-A Pair122
Q-A Pair123
Q-A Pair1n1
Q-A Pair1n2
Q-A Pair1n3
Q-A Pairnn1
Q-A Pairnn2
Q-A Pairnnn

Fig. 7A

FIELD NAME 701A	DATA TYPE 702A	SIZE 703A	NULL 704A	PRIMARY KEY 705A	INDEXED? 706A
ChapterName 707A	Varchar	255	No	No	Yes
SectionName 708A	Varchar	255	No	No	Yes

Fig.7B

FIELD NAME 720	DATA TYPE 721	SIZE 722	NULL 723	PRIMARY KEY 724	INDEXED? 725
Chapter_ID 726	Integer		No	Yes	Yes
Answer_ID 727	Char	5	No	UNIQUE	Yes
Section_Name 728	Varchar	255	No	UNIQUE	Yes
Answer_Title 729	Varchar	255	Yes	No	Yes
PairedQuestion 730	Text	16	No	No	Yes (Full-Text)
AnswerPath 731	Varchar	255	No	No	Yes
Creator 732	Varchar	50	No	No	Yes
Date_of_Creation 733	Date	-	No	No	Yes
Date_of_Modification 734	Date	-	No	No	Yes

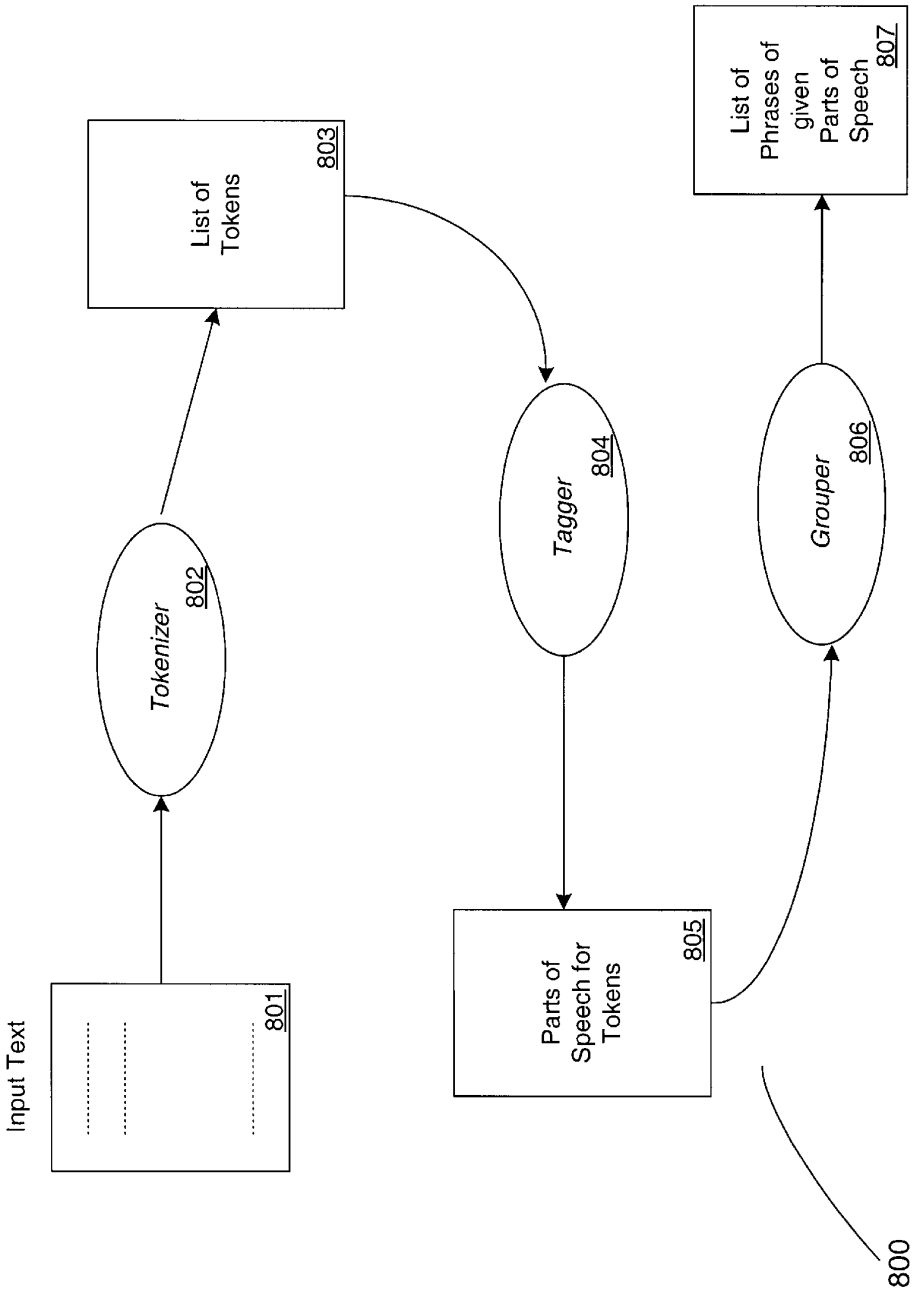
Fig. 7C

Field 720	Description 735
AnswerID 727	An integer - automatically incremented for user convenience
Section_Name 728	Name of section to which the particular record belongs. This field along with AnswerID has to be made primary key
Answer_Title 729	A short description of the answer
PairedQuestion 730	Contains one or more combinations of questions for the related answer whose path is stored in the next column AnswerPath
AnswerPath 731	Contains the path of text file, which contains the answer to the related questions stored in the previous column
Creator 732	Name of content creator
Date_of_Creation 733	Date on which content has been added
Date_of_Modification 734	Date on which content has been changed or modified

Fig. 7D

Field 740	Data Type 741	Size 742	Null 743	Primary Key 744	Indexed 745
Answer_ID 746	Char	5	No	Yes	Yes
Answer_Title 747	Varchar	255	Yes	No	No
PairedQuestion 748	Text	16	No	No	Yes (Full-Text)
Answer_Path 749	Varchar	255	No	No	No
Creator 750	Varchar	50	No	No	No
Date_of_Creation 751	Date	-	No	No	No
Date_of_Modification 752	Date	-	No	No	No

Fig. 8


Input Text
801
Tokenizer
802
List of Tokens
803
Tagger
804
Parts of Speech for Tokens
805
Grouper
806
List of Phrases of given Parts of Speech
807
800

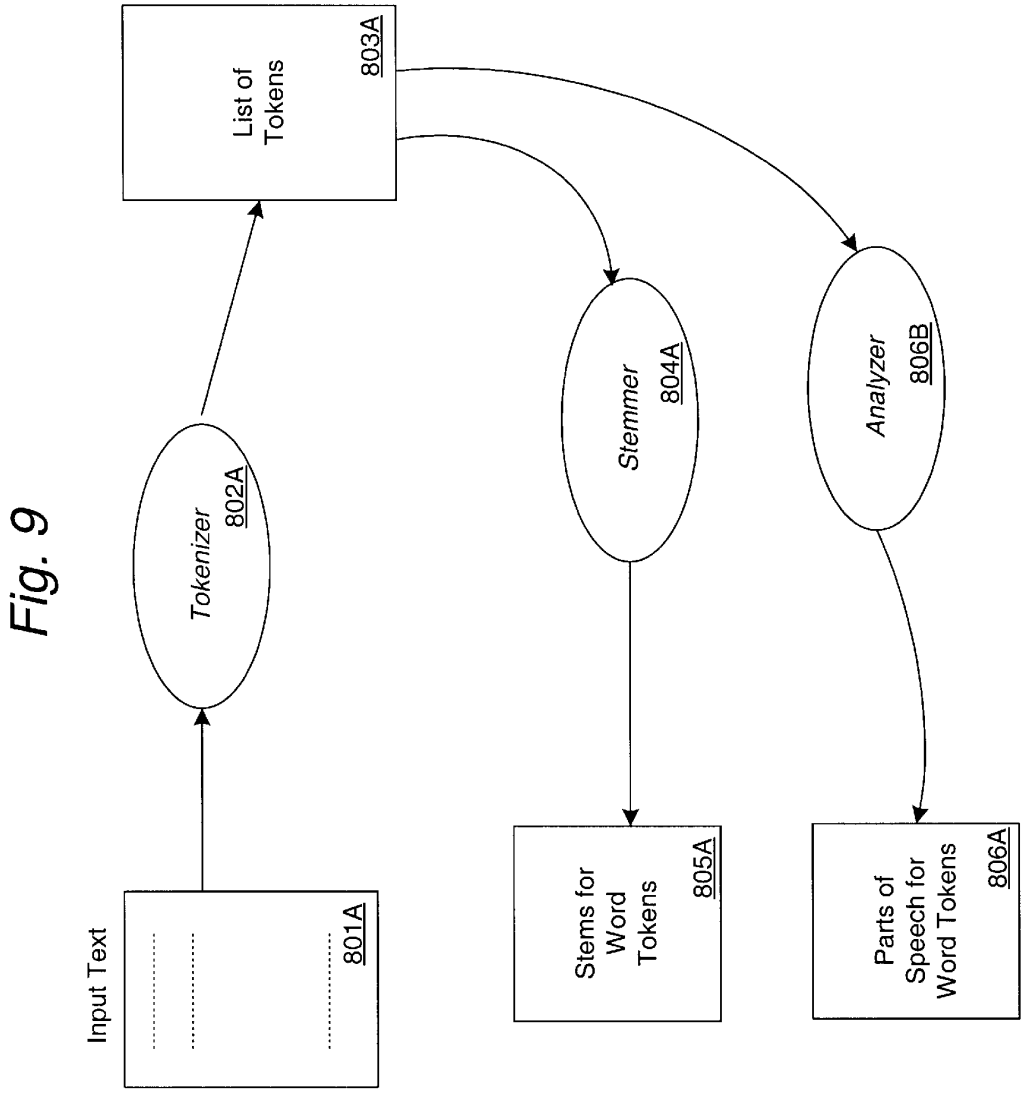

Input Text
801A
Tokenizer
802A
List of Tokens
803A
Stemmer
804A
Stems for Word Tokens
805A
Analyzer
806B
Parts of Speech for Word Tokens
806A

Fig. 9

Fig. 10

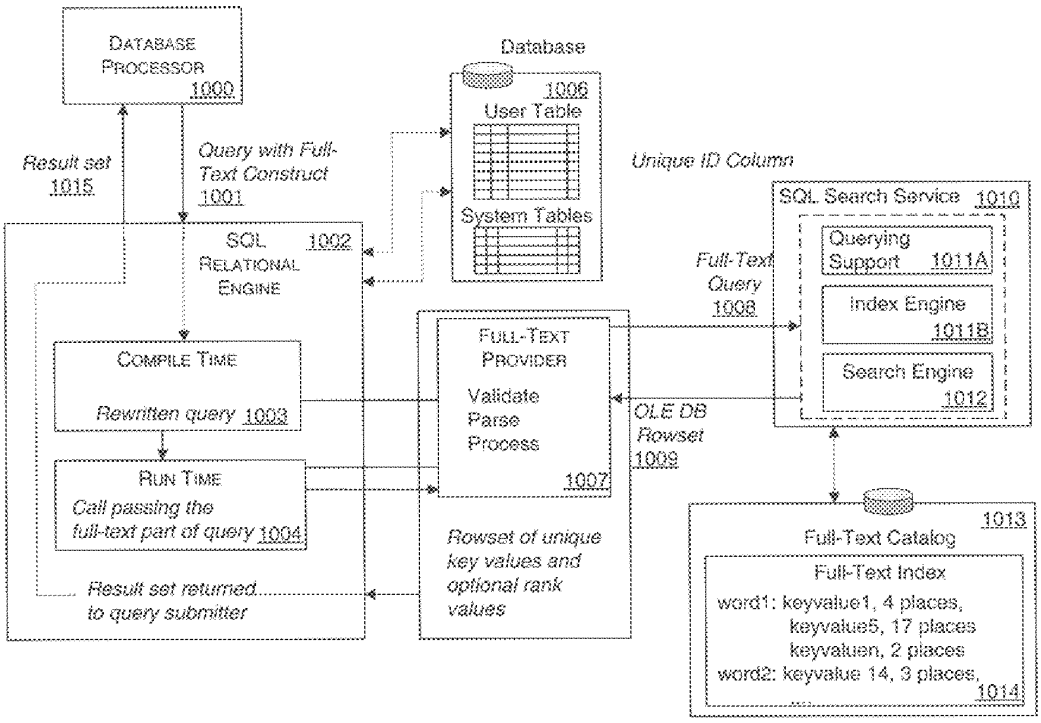

DATABASE PROCESSOR 1000
Database
1006
User Table
System Tables
Result set 1015
Query with Full-Text Construct 1001
Unique ID Column
SQL Search Service 1010
Querying Support 1011A
Index Engine 1011B
Search Engine 1012
SQL RELATIONAL ENGINE 1002
Full-Text Query 1008
COMPILE TIME
Rewritten query 1003
FULL-TEXT PROVIDER
Validate
Parse
Process
1007
OLE DB Rowset 1009
RUN TIME
Call passing the full-text part of query 1004
Rowset of unique key values and optional rank values
Result set returned to query submitter
1013
Full-Text Catalog
Full-Text Index
word1: keyvalue1, 4 places,
keyvalue5, 17 places
keyvaluen, 2 places
word2: keyvalue 14, 3 places,
1014

Fig. 11A

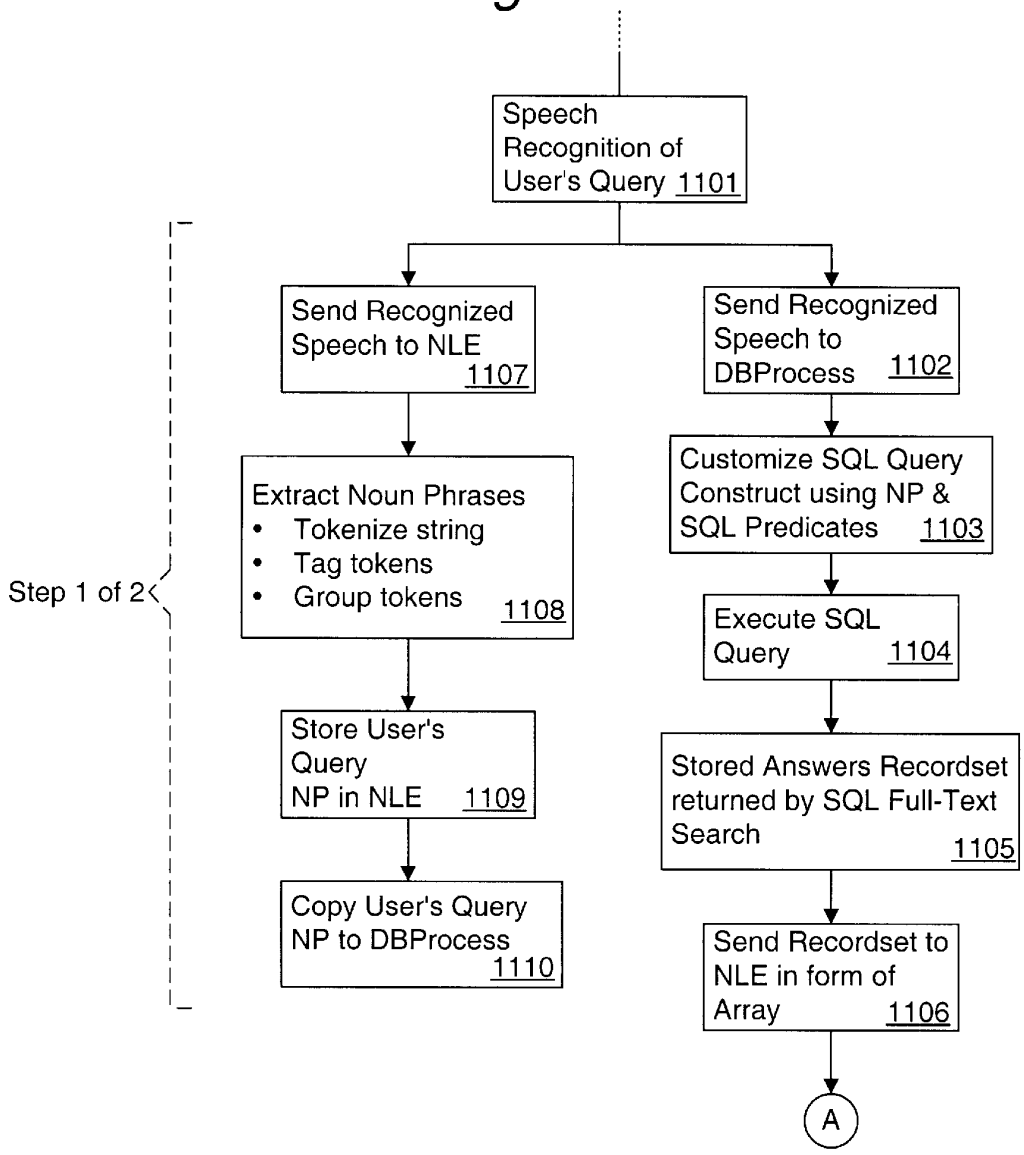

Speech Recognition of User's Query 1101
Send Recognized Speech to NLE 1107
Send Recognized Speech to DBProcess 1102
Extract Noun Phrases
• Tokenize string
• Tag tokens
• Group tokens 1108
Customize SQL Query Construct using NP & SQL Predicates 1103
Execute SQL Query 1104
Store User's Query NP in NLE 1109
Stored Answers Recordset returned by SQL Full-Text Search 1105
Copy User's Query NP to DBProcess 1110
Send Recordset to NLE in form of Array 1106
A
Step 1 of 2

Fig. 11B

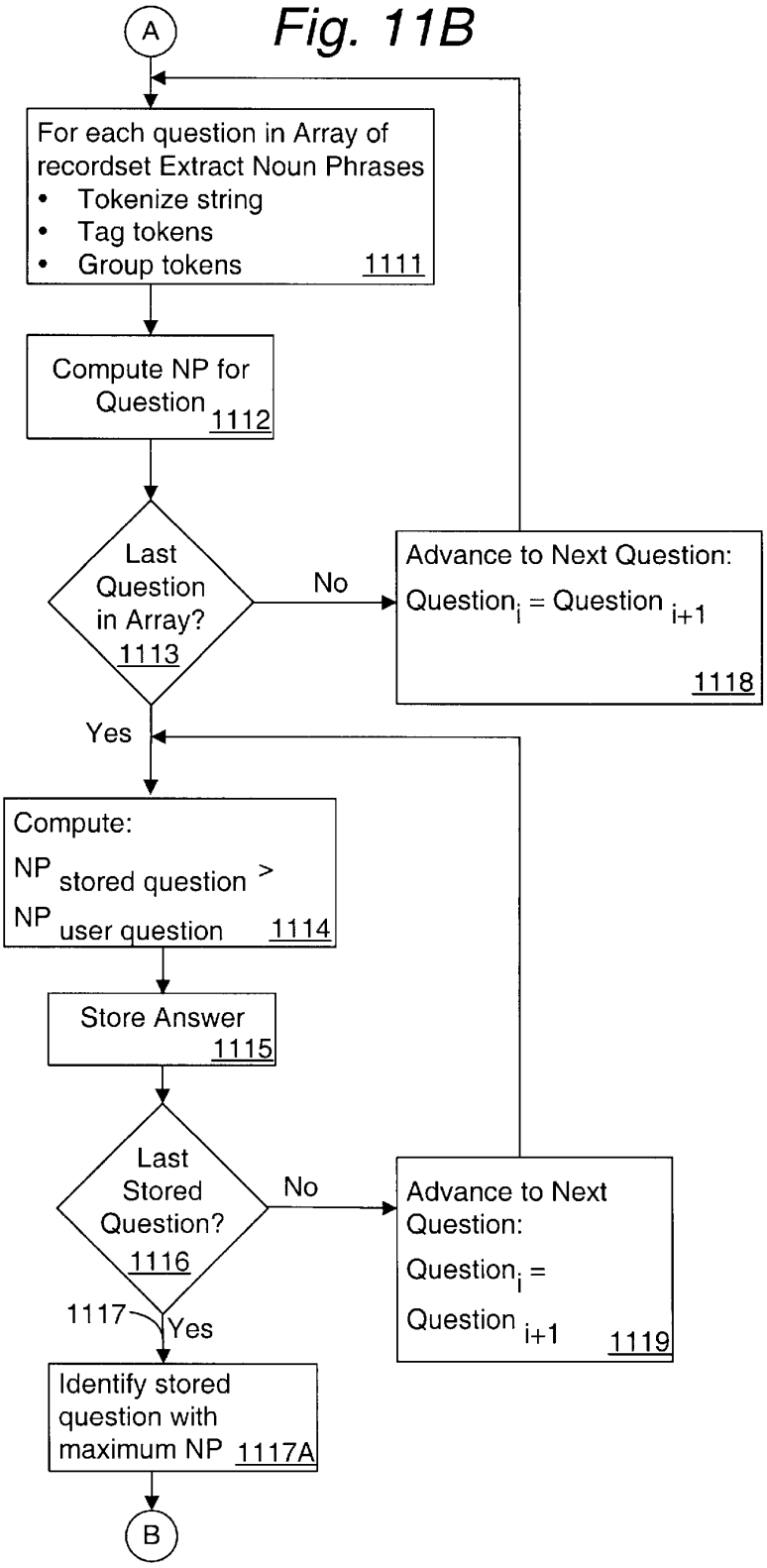

A
For each question in Array of recordset Extract Noun Phrases
• Tokenize string
• Tag tokens
• Group tokens 1111
Compute NP for Question 1112
Last Question in Array? 1113
No
Advance to Next Question: Question$_i$ = Question $_{i+1}$ 1118
Yes
Compute: NP stored question > NP user question 1114
Store Answer 1115
Last Stored Question? 1116
No
Advance to Next Question: Question$_i$ = Question $_{i+1}$ 1119
1117
Yes
Identify stored question with maximum NP 1117A
B

Fig. 11C

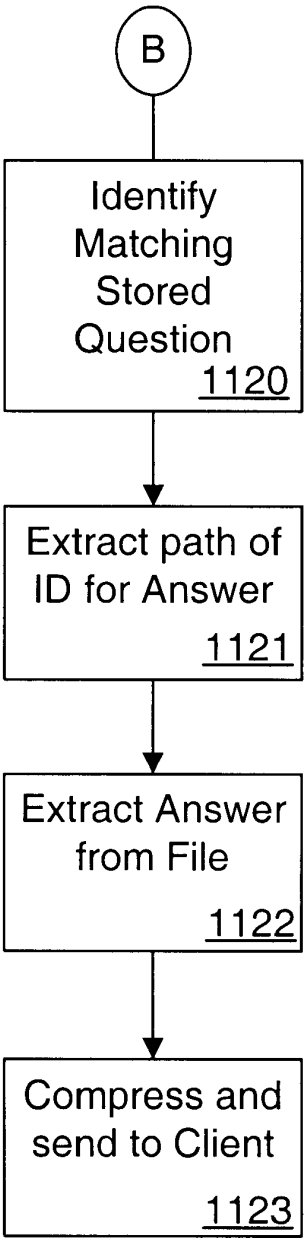

B
Identify Matching Stored Question
1120
Extract path of ID for Answer
1121
Extract Answer from File
1122
Compress and send to Client
1123

Fig. 12

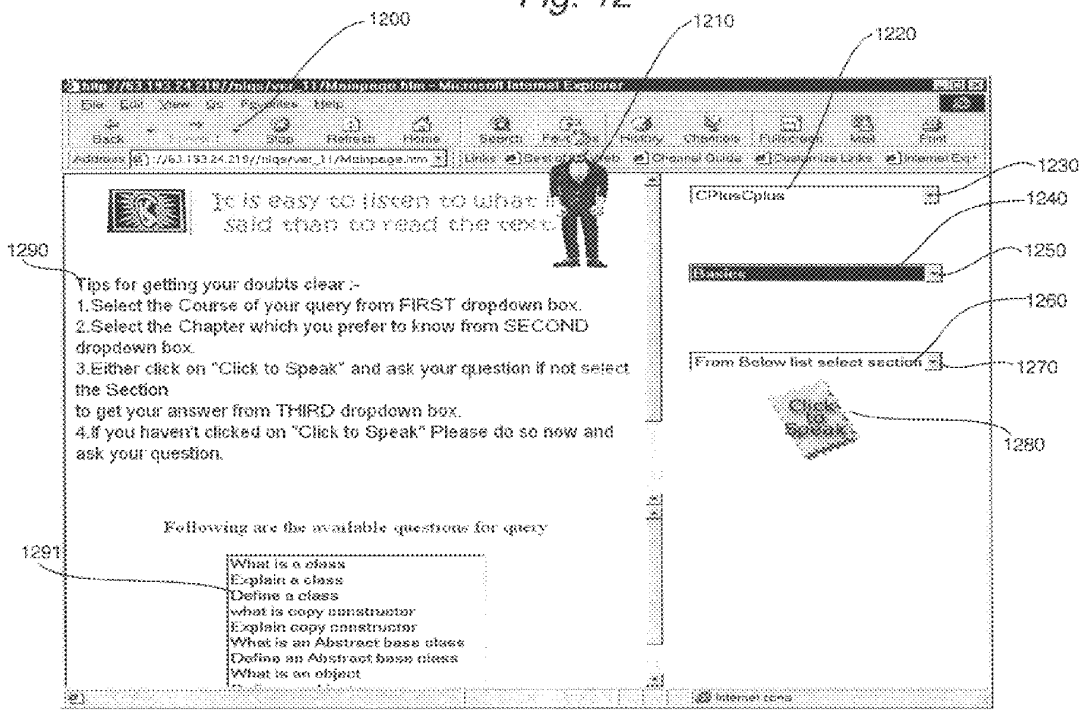

1200
1210
1220
1230
1240
1250
1260
1270
1280
1290
1291
CPlusCplus
From Below list select section
Tips for getting your doubts clear :-
1.Select the Course of your query from FIRST dropdown box.
2.Select the Chapter which you prefer to know from SECOND dropdown box.
3.Either click on "Click to Speak" and ask your question if not select the Section
to get your answer from THIRD dropdown box.
4.If you haven't clicked on "Click to Speak" Please do so now and ask your question.
Following are the available questions for query
What is a class
Explain a class
Define a class
what is copy constructor
Explain copy constructor
What is an Abstract base class
Define an Abstract base class
What is an object

Fig. 13

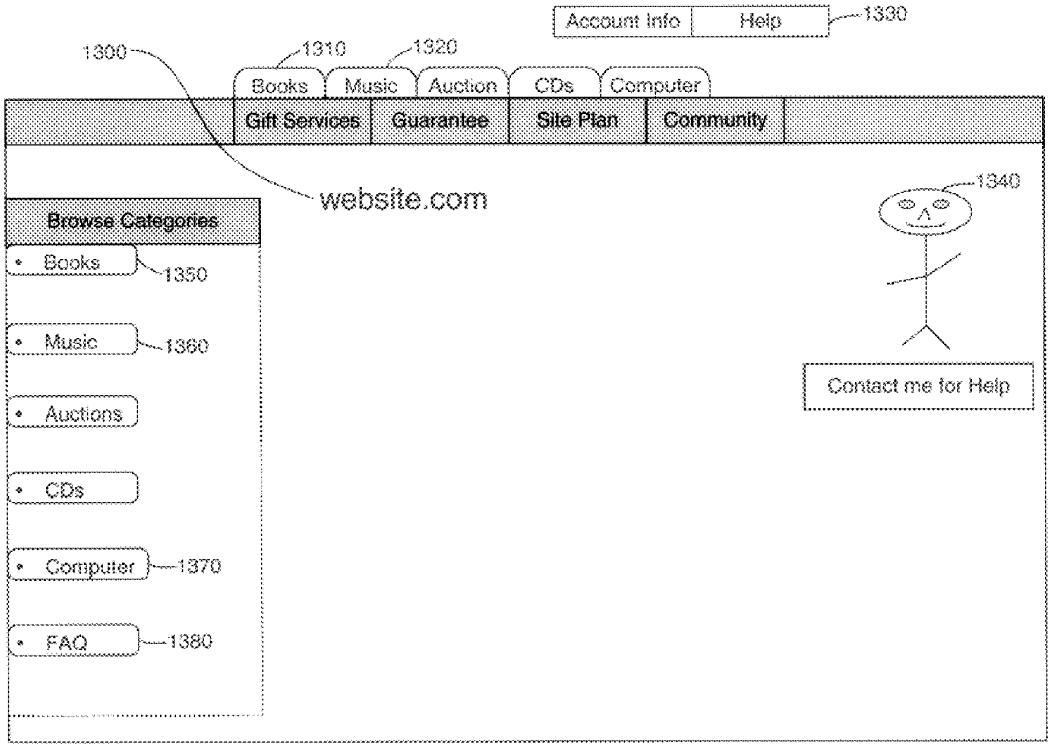

Account Info
Help
1330
1300
1310
1320
Books
Music
Auction
CDs
Computer
Gift Services
Guarantee
Site Plan
Community
website.com
1340
Browse Categories
Books
1350
Music
1360
Auctions
CDs
Computer
1370
FAQ
1380
Contact me for Help

Fig. 14

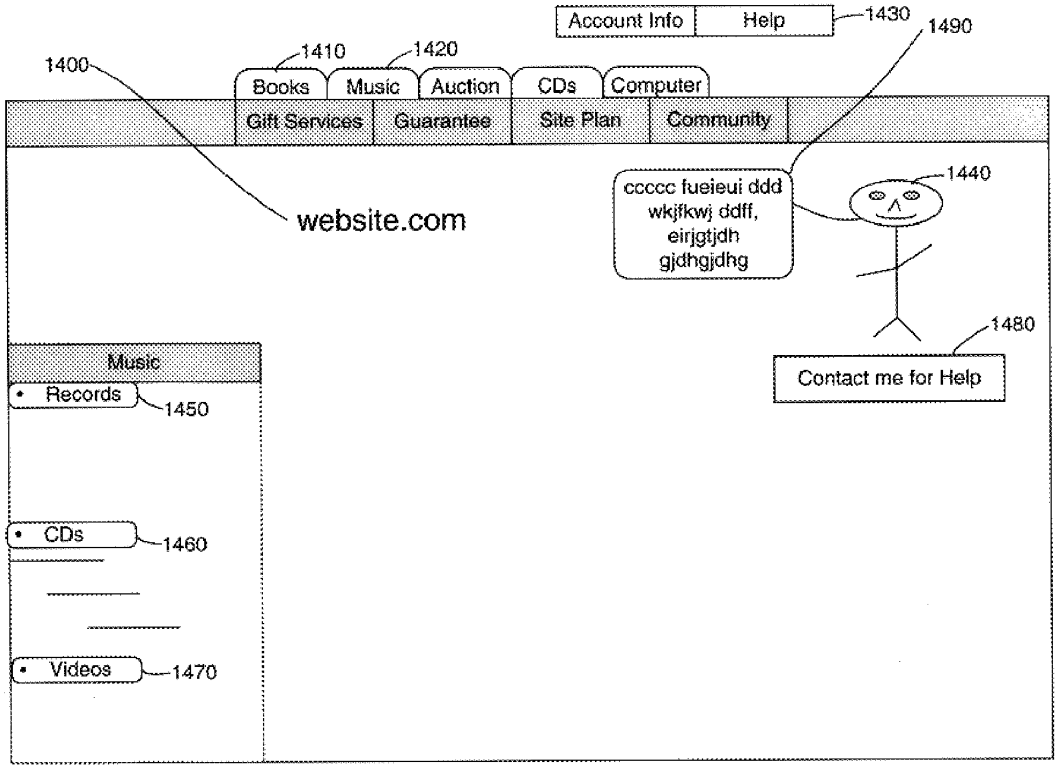

Account Info
Help
1430
1490
1400
1410
1420
Books
Music
Auction
CDs
Computer
Gift Services
Guarantee
Site Plan
Community
website.com
ccccc fueieui ddd
wkjfkwj ddff,
eirjgtjdh
gjdhgjdhg
1440
1480
Contact me for Help
Music
Records
1450
CDs
1460
Videos
1470

Fig. 15

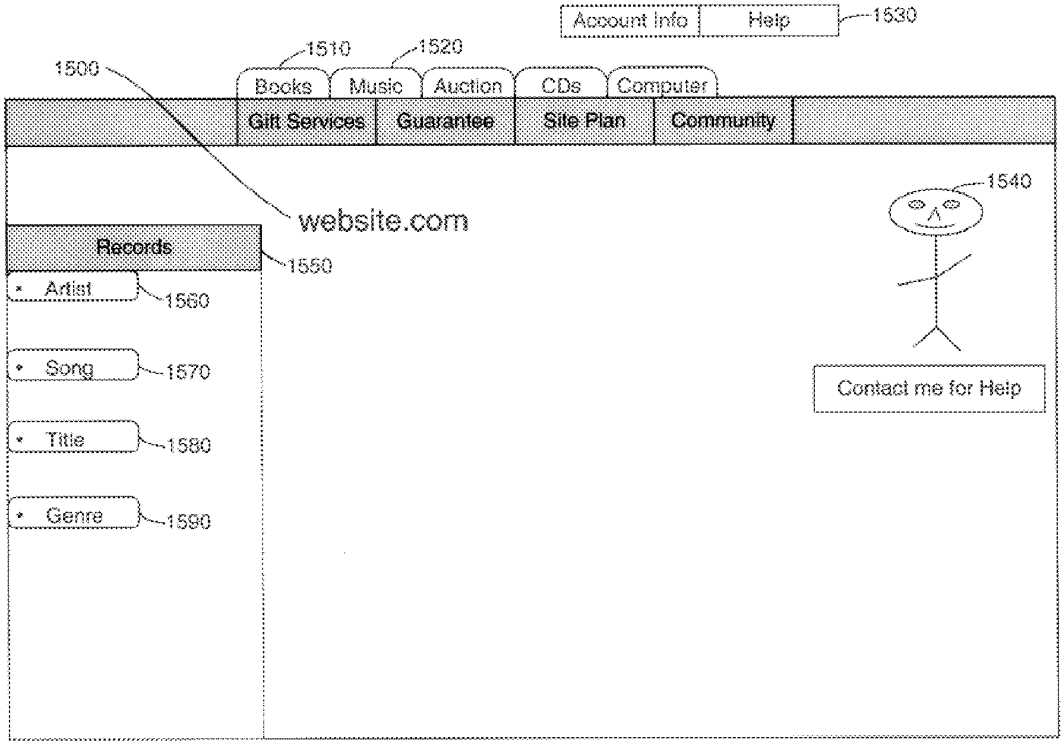

Account Info
Help
1530
1510
1520
1500
Books
Music
Auction
CDs
Computer
Gift Services
Guarantee
Site Plan
Community
website.com
1540
Records
1550
Artist
1560
Song
1570
Title
1580
Genre
1590
Contact me for Help

Fig. 16

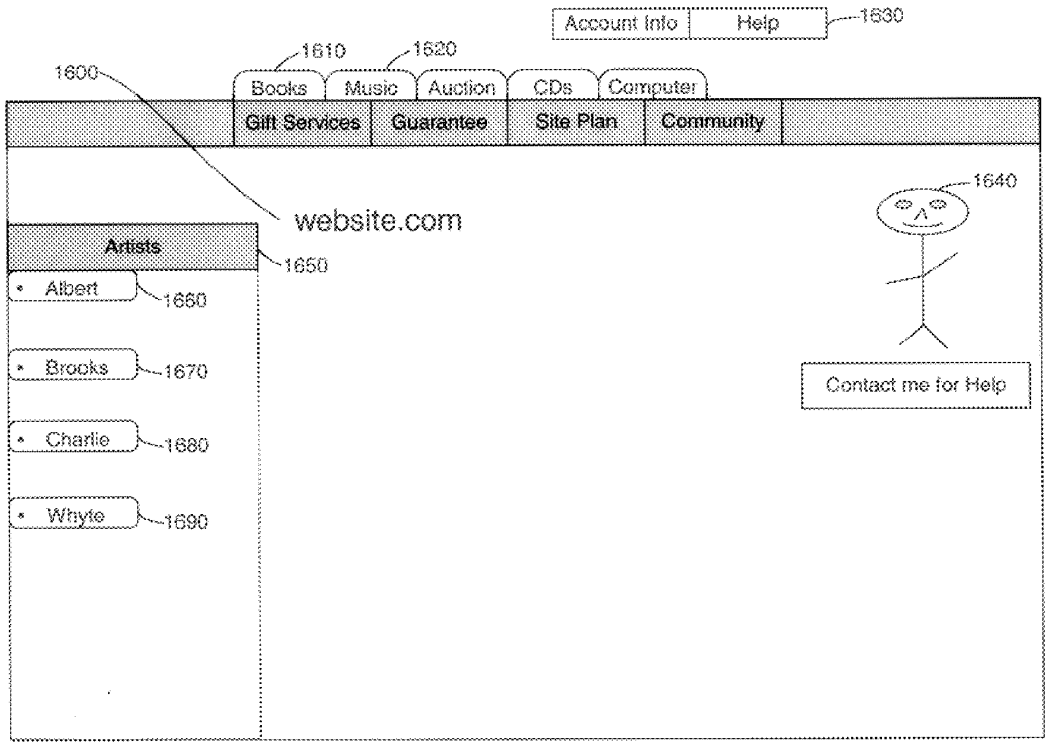

Account Info
Help
1630
1610
1620
1600
Books
Music
Auction
CDs
Computer
Gift Services
Guarantee
Site Plan
Community
website.com
1640
Artists
1650
Albert
1660
Brooks
1670
Charlie
1680
Whyte
1690
Contact me for Help

Fig. 17

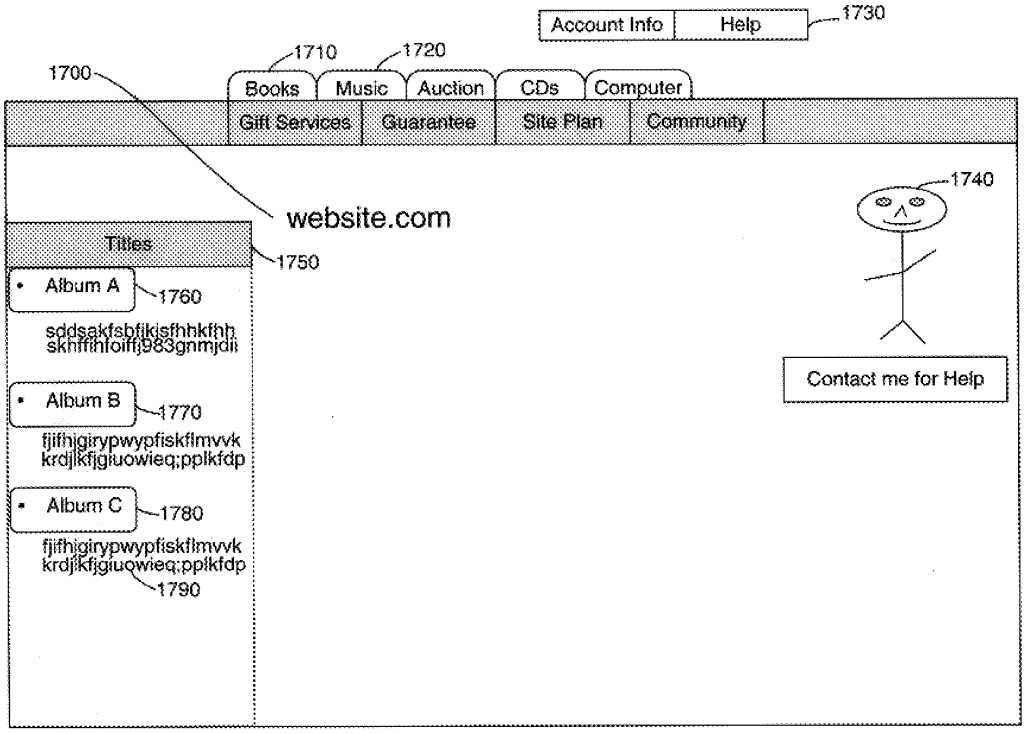

Account Info
Help
1730
1710
1720
1700
Books
Music
Auction
CDs
Computer
Gift Services
Guarantee
Site Plan
Community
website.com
1740
Titles
1750
Album A
1760
sddsakfsbfjkisfhhkfhh
skhffihfoiffj983gnmjdii
Album B
1770
fjifhigirypwypfiskflmvvk
krdjlkfjgiuowieq;pplkfdp
Album C
1780
fjifhigirypwypfiskflmvvk
krdjlkfjgiuowieq;pplkfdp
1790
Contact me for Help

Fig. 18A

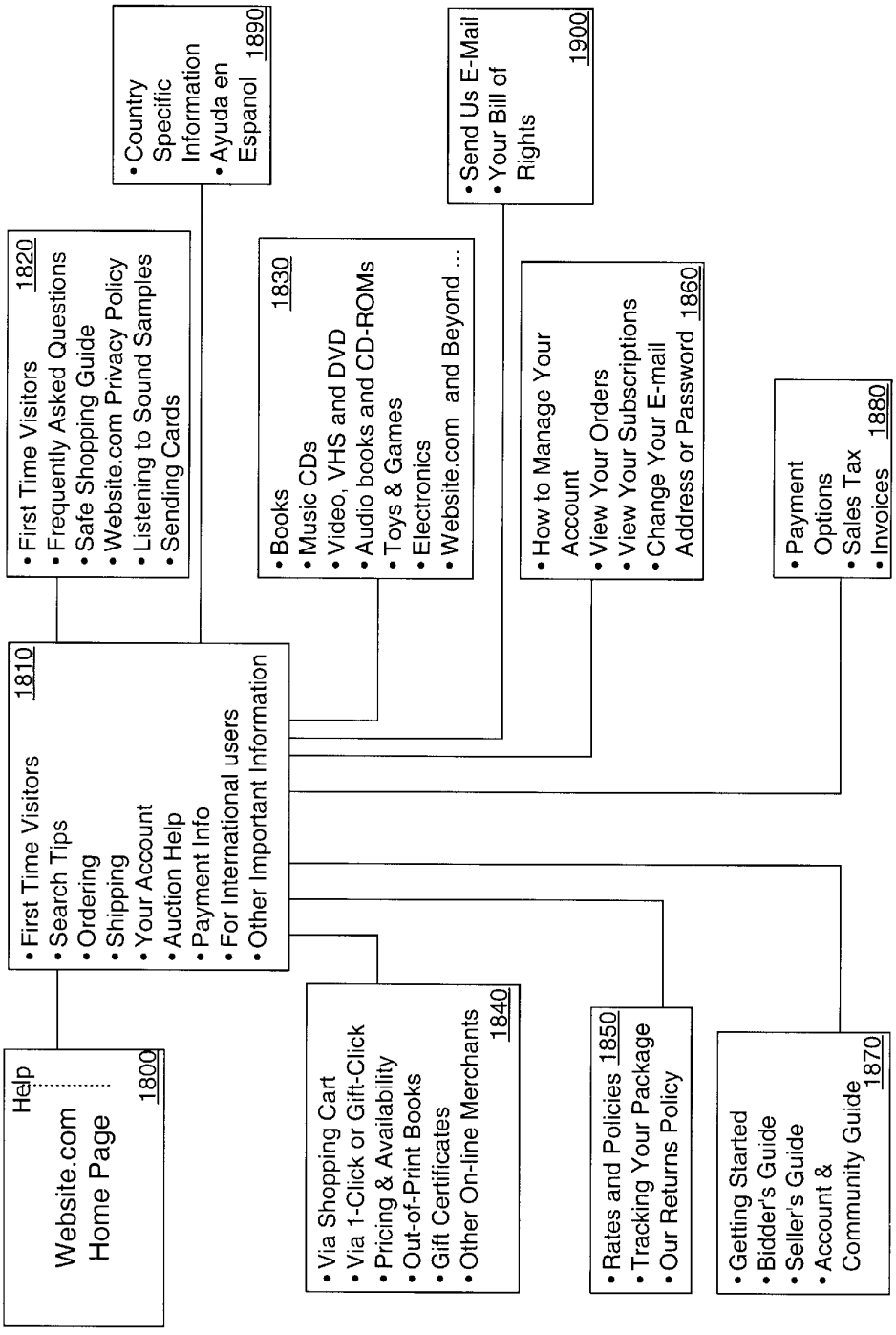
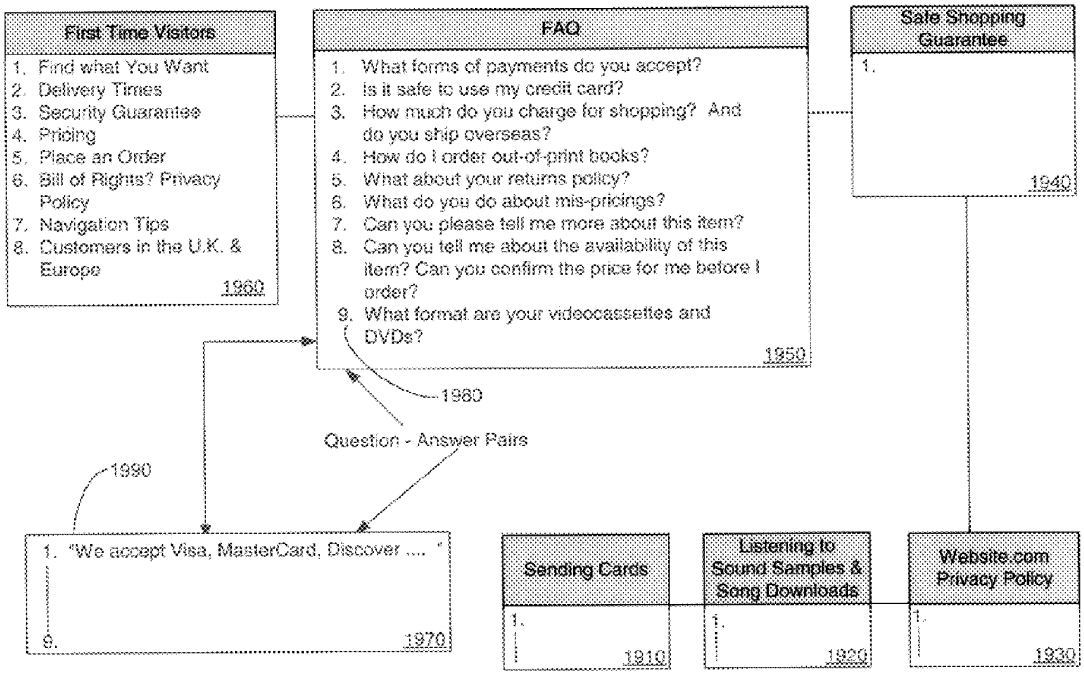

Website.com Home Page
Help
1800
First Time Visitors
Search Tips
Ordering
Shipping
Your Account
Auction Help
Payment Info
For International users
Other Important Information
1810
First Time Visitors
Frequently Asked Questions
Safe Shopping Guide
Website.com Privacy Policy
Listening to Sound Samples
Sending Cards
1820
Country Specific Information
Ayuda en Espanol
1890
Books
Music CDs
Video, VHS and DVD
Audio books and CD-ROMs
Toys & Games
Electronics
Website.com and Beyond ...
1830
Send Us E-Mail
Your Bill of Rights
1900
How to Manage Your Account
View Your Orders
View Your Subscriptions
Change Your E-mail Address or Password
1860
Payment Options
Sales Tax
Invoices
1880
Via Shopping Cart
Via 1-Click or Gift-Click
Pricing & Availability
Out-of-Print Books
Gift Certificates
Other On-line Merchants
1840
Rates and Policies
1850
Tracking Your Package
Our Returns Policy
Getting Started
Bidder's Guide
Seller's Guide
Account & Community Guide
1870

Fig. 18B


First Time Visitors
1. Find what You Want
2. Delivery Times
3. Security Guarantee
4. Pricing
5. Place an Order
6. Bill of Rights? Privacy Policy
7. Navigation Tips
8. Customers in the U.K. & Europe
1960
FAQ
1. What forms of payments do you accept?
2. Is it safe to use my credit card?
3. How much do you charge for shopping? And do you ship overseas?
4. How do I order out-of-print books?
5. What about your returns policy?
6. What do you do about mis-pricings?
7. Can you please tell me more about this item?
8. Can you tell me about the availability of this item? Can you confirm the price for me before I order?
9. What format are your videocassettes and DVDs?
1950
Safe Shopping Guarantee
1.
1940
1980
Question - Answer Pairs
1990
1. "We accept Visa, MasterCard, Discover ...."
9.
1970
Sending Cards
1.
1910
Listening to Sound Samples & Song Downloads
1.
1920
Website.com Privacy Policy
1.
1930

DISTRIBUTED REALTIME SPEECH RECOGNITION SYSTEM

RELATED APPLICATIONS

The present application is related to the following applications also filed contemporaneously herewith:

1) Ser. No. 09/439,173 entitled Speech Based Learning/Training system, attorney docket no. PHO 99-002;

2) Ser. No. 09/439,174 entitled Internet Server with Speech Support for Enhanced Interactivity—attorney docket no. PHO 99-003;

3) Ser. No. 09/439,060 entitled Intelligent Query Engine For Processing Voice Based Queries—attorney docket no. PHO 99-004;

The above are incorporated by reference herein.

FIELD OF THE INVENTION

The invention relates to a system and an interactive method for responding to speech based user inputs and queries presented over a distributed network such as the INTERNET or local intranet. This interactive system when implemented over the World-Wide Web services (WWW) of the INTERNET, functions so that a client or user can ask a question in a natural language such as English, French, German, Spanish or Japanese and receive the appropriate answer at his or her computer or accessory also in his or her native natural language. The system has particular applicability to such applications as remote learning, e-commerce, technical e-support services, INTERNET searching, etc.

BACKGROUND OF THE INVENTION

The INTERNET, and in particular, the World-Wide Web (WWW, is growing in popularity and usage for both commercial and recreational purposes, and this trend is expected to continue. This phenomenon is being driven, in part, by the increasing and widespread use of personal computer systems and the availability of low cost INTERNET access. The emergence of inexpensive INTERNET access devices and high speed access techniques such as ADSL, cable modems, satellite modems, and the like, are expected to further accelerate the mass usage of the WWW.

Accordingly, it is expected that the number of entities offering services, products, etc., over the WWW will increase dramatically over the coming years. Until now, however, the INTERNET "experience" for users has been limited mostly to non-voice based input/output devices, such as keyboards, intelligent electronic pads, mice, trackballs, printers, monitors, etc. This presents somewhat of a bottleneck for interacting over the WWW for a variety of reasons.

First, there is the issue of familiarity. Many kinds of applications lend themselves much more naturally and fluently to a voice-based environment. For instance, most people shopping for audio recordings are very comfortable with asking a live sales clerk in a record store for information on titles by a particular author, where they can be found in the store, etc. While it is often possible to browse and search on one's own to locate items of interest, it is usually easier and more efficient to get some form of human assistance first, and, with few exceptions, this request for assistance is presented in the form of a oral query. In addition, many persons cannot or will not, because of physical or psychological barriers, use any of the aforementioned conventional I/O devices. For example, many older persons cannot easily read the text presented on WWW pages, or understand the layout/hierarchy of menus, or

manipulate a mouse to make finely coordinated movements to indicate their selections. Many others are intimidated by the look and complexity of computer systems, WWW pages, etc., and therefore do not attempt to use online services for this reason as well.

Thus, applications which can mimic normal human interactions are likely to be preferred by potential on-line shoppers and persons looking for information over the WWW. It is also expected that the use of voice-based systems will increase the universe of persons willing to engage in e-commerce, e-learning, etc. To date, however, there are very few systems, if any, which permit this type of interaction, and, if they do, it is very limited. For example, various commercial programs sold by IBM (VIAVOICE™) and Kurzweil (DRAGON™) permit some user control of the interface (opening, closing files) and searching (by using previously trained URLs) but they do not present a flexible solution that can be used by a number of users across multiple cultures and without time consuming voice training. Typical prior efforts to implement voice based functionality in an INTERNET context can be seen in U.S. Pat. No. 5,819,220 incorporated by reference herein.

Another issue presented by the lack of voice-based systems is efficiency. Many companies are now offering technical support over the INTERNET, and some even offer live operator assistance for such queries. While this is very advantageous (for the reasons mentioned above) it is also extremely costly and inefficient, because a real person must be employed to handle such queries. This presents a practical limit that results in long wait times for responses or high labor overheads. An example of this approach can be seen U.S. Pat. No. 5,802,526 also incorporated by reference herein. In general, a service presented over the WWW is far more desirable if it is "scalable," or, in other words, able to handle an increasing amount of user traffic with little if any perceived delay or troubles by a prospective user.

In a similar context, while remote learning has become an increasingly popular option for many students, it is practically impossible for an instructor to be able to field questions from more than one person at a time. Even then, such interaction usually takes place for only a limited period of time because of other instructor time constraints. To date, however, there is no practical way for students to continue a human-like question and answer type dialog after the learning session is over, or without the presence of the instructor to personally address such queries.

Conversely, another aspect of emulating a human-like dialog involves the use of oral feedback. In other words, many persons prefer to receive answers and information in audible form. While a form of this functionality is used by some websites to communicate information to visitors, it is not performed in a real-time, interactive question-answer dialog fashion so its effectiveness and usefulness is limited.

Yet another area that could benefit from speech-based interaction involves so-called "search" engines used by INTERNET users to locate information of interest at web sites, such as the those available at YAHOO®.com, METACRAWLER®.com, EXCITE®.com, etc. These tools permit the user to form a search query using either combinations of keywords or metacategories to search through a web page database containing text indices associated with one or more distinct web pages. After processing the user's request, therefore, the search engine returns a number of hits which correspond, generally, to URL pointers and text excerpts from the web pages that represent the closest match made by such search engine for the particular user query

based on the search processing logic used by search engine. The structure and operation of such prior art search engines, including the mechanism by which they build the web page database, and parse the search query, are well known in the art. To date, applicant is unaware of any such search engine that can easily and reliably search and retrieve information based on speech input from a user.

There are a number of reasons why the above environments (e-commerce, e-support, remote learning, INTERNET searching, etc.) do not utilize speech-based interfaces, despite the many benefits that would otherwise flow from such capability. First, there is obviously a requirement that the output of the speech recognizer be as accurate as possible. One of the more reliable approaches to speech recognition used at this time is based on the Hidden Markov Model (HMM)—a model used to mathematically describe any time series. A conventional usage of this technique is disclosed, for example, in U.S. Pat. No. 4,587,670 incorporated by reference herein. Because speech is considered to have an underlying sequence of one or more symbols, the HMM models corresponding to each symbol are trained on vectors from the speech waveforms. The Hidden Markov Model is a finite set of states, each of which is associated with a (generally multi-dimensional) probability distribution. Transitions among the states are governed by a set of probabilities called transition probabilities. In a particular state an outcome or observation can be generated, according to the associated probability distribution. This finite state machine changes state once every time unit, and each time t such that a state j is entered, a spectral parameter vector O_t is generated with probability density $B_j(O_t)$. It is only the outcome, not the state visible to an external observer and therefore states are "hidden" to the outside; hence the name Hidden Markov Model. The basic theory of HMMs was published in a series of classic papers by Baum and his colleagues in the late 1960's and early 1970's. HMMs were first used in speech applications by Baker at Carnegie Mellon, by Jelenik and colleagues at IBM in the late 1970's and by Steve Young and colleagues at Cambridge University, UK in the 1990's. Some typical papers and texts are as follows:

1. L. E. Baum, T. Petrie, "Statistical inference for probabilistic functions for finite state Markov chains", Ann. Math. Stat., 37: 1554–1563, 1966
2. L. E. Baum, "An inequality and associated maximation technique in statistical estimation for probabilistic functions of Markov processes", Inequalities 3: 1–8, 1972
3. J. H. Baker, "The dragon system—An Overview", IEEE Trans. on ASSP Proc., ASSP-23(1): 24–29, Feb. 1975
4. F. Jeninek et al, "Continuous Speech Recognition: Statistical methods" in Handbook of Statistics, II, P. R. Kristnaiad, Ed. Amsterdam, The Netherlands, North-Holland, 1982
5. L. R. Bahl, F. Jeninek, R. L. Mercer, "A maximum likelihood approach to continuous speech recognition", IEEE Trans. Pattern Anal. Mach. Intell., PAMI-5: 179–190,1983
6. J. D. Ferguson, "Hidden Markov Analysis: An Introduction", in Hidden Markov Models for Speech, Institute of Defense Analyses, Princeton, N.J. 1980.
7. H. R. Rabiner and B. H. Juang, "Fundamentals of Speech Recognition", Prentice Hall, 1993
8. H. R. Rabiner, "Digital Processing of Speech Signals", Prentice Hall, 1978

More recently research has progressed in extending HMM and combining HMMs with neural networks to speech recognition applications at various laboratories. The following is a representative paper:

9. Nelson Morgan, Hervé Bourlard, Steve Renals, Michael Cohen and Horacio Franco (1993), Hybrid Neural Network/Hidden Markov Model Systems for Continuous Speech Recognition. *Journal of Pattern Recognition and Artificial Intelligence,* Vol.7, No.4 pp. 899–916. Also in I. Guyon and P. Wang editors, *Advances in Pattern Recognition Systems using Neural Networks,* Vol. 7 of a Series in Machine Perception and Artificial Intelligence. World Scientific, Feb. 1994.

All of the above are hereby incorporated by reference. While the HMM-based speech recognition yields very good results, contemporary variations of this technique cannot guarantee a word accuracy requirement of 100% exactly and consistently, as will be required for WWW applications for all possible all user and environment conditions. Thus, although speech recognition technology has been available for several years, and has improved significantly, the technical requirements have placed severe restrictions on the specifications for the speech recognition accuracy that is required for an application that combines speech recognition and natural language processing to work satisfactorily.

In contrast to word recognition, Natural language processing (NLP) is concerned with the parsing, understanding and indexing of transcribed utterances and larger linguistic units. Because spontaneous speech contains many surface phenomena such as disfluencies,—hesitations, repairs and restarts, discourse markers such as 'well' and other elements which cannot be handled by the typical speech recognizer, it is the problem and the source of the large gap that separates speech recognition and natural language processing technologies. Except for silence between utterances, another problem is the absence of any marked punctuation available for segmenting the speech input into meaningful units such as utterances. For optimal NLP performance, these types of phenomena should be annotated at its input. However, most continuous speech recognition systems produce only a raw sequence of words. Examples of conventional systems using NLP are shown in U.S. Pat. Nos. 4,991,094, 5,068,789, 5,146,405 and 5,680,628, all of which are incorporated by reference herein.

Second, most of the very reliable voice recognition systems are speaker-dependent, requiring that the interface be "trained" with the user's voice, which takes a lot of time, and is thus very undesirable from the perspective of a WWW environment, where a user may interact only a few times with a particular website. Furthermore, speaker-dependent systems usually require a large user dictionary (one for each unique user) which reduces the speed of recognition. This makes it much harder to implement a real-time dialog interface with satisfactory response capability (i.e., something that mirrors normal conversation—on the order of 3–5 seconds is probably ideal). At present, the typical shrink-wrapped speech recognition application software include offerings from IBM (VIAVOICE™) and Dragon Systems (DRAGON™). While most of these applications are adequate for dictation and other transcribing applications, they are woefully inadequate for applications such as NLQS where the word error rate must be close to 0%. In addition these offerings require long training times and are typically are non client-server configurations. Other types of trained systems are discussed in U.S. Pat. No. 5,231,670 assigned to Kurzweil, and which is also incorporated by reference herein.

Another significant problem faced in a distributed voice-based system is a lack of uniformity/control in the speech recognition process. In a typical stand-alone implementation of a speech recognition system, the entire SR engine runs on a single client. A well-known system of this type is depicted in U.S. Pat. No. 4,991,217 incorporated by reference herein. These clients can take numerous forms (desktop PC, laptop PC, PDA, etc.) having varying speech signal processing and communications capability. Thus, from the server side perspective, it is not easy to assure uniform treatment of all users accessing a voice-enabled web page, since such users may have significantly disparate word recognition and error rate performances. While a prior art reference to Gould et al.—U.S. Pat. No. 5,915,236—discusses generally the notion of tailoring a recognition process to a set of available computational resources, it does not address or attempt to solve the issue of how to optimize resources in a distributed environment such as a client-server model. Again, to enable such voice-based technologies on a wide-spread scale it is far more preferable to have a system that harmonizes and accounts for discrepancies in individual systems so that even the thinnest client is supportable, and so that all users are able to interact in a satisfactory manner with the remote server running the e-commerce, e-support and/or remote learning application.

Two references that refer to a distributed approach for speech recognition include U.S. Pat. Nos. 5,956,683 and 5,960,399 incorporated by reference herein. In the first of these, U.S. Pat. No. 5,956,683—Distributed Voice Recognition System (assigned to Qualcomm) an implementation of a distributed voice recognition system between a telephony-based handset and a remote station is described. In this implementation, all of the word recognition operations seem to take place at the handset. This is done since the patent describes the benefits that result from locating of the system for acoustic feature extraction at the portable or cellular phone in order to limit degradation of the acoustic features due to quantization distortion resulting from the narrow bandwidth telephony channel. This reference therefore does not address the issue of how to ensure adequate performance for a very thin client platform. Moreover, it is difficult to determine, how, if at all, the system can perform real-time word recognition, and there is no meaningful description of how to integrate the system with a natural language processor.

The second of these references—U.S. Pat. No. 5,960,399—Client/Server Speech Processor/Recognizer (assigned to GTE) describes the implementation of a HMM-based distributed speech recognition system. This reference is not instructive in many respects, however, including how to optimize acoustic feature extraction for a variety of client platforms, such as by performing a partial word recognition process where appropriate. Most importantly, there is only a description of a primitive server-based recognizer that only recognizes the user's speech and simply returns certain keywords such as the user's name and travel destination to fill out a dedicated form on the user's machine. Also, the streaming of the acoustic parameters does not appear to be implemented in real-time as it can only take place after silence is detected. Finally, while the reference mentions the possible use of natural language processing (column 9) there is no explanation of how such function might be implemented in a real-time fashion to provide an interactive feel for the user.

SUMMARY OF THE INVENTION

An object of the present invention, therefore, is to provide an improved system and method for overcoming the limitations of the prior art noted above;

A primary object of the present invention is to provide a word and phrase recognition system that is flexibly and optimally distributed across a client/platform computing architecture, so that improved accuracy, speed and uniformity can be achieved for a wide group of users;

A further object of the present invention is to provide a speech recognition system that efficiently integrates a distributed word recognition system with a natural language processing system, so that both individual words and entire speech utterances can be quickly and accurately recognized in any number of possible languages;

A related object of the present invention is to provide an efficient query response system so that an extremely accurate, real-time set of appropriate answers can be given in response to speech-based queries;

Yet another object of the present invention is to provide an interactive, real-time instructional/learning system that is distributed across a client/server architecture, and permits a real-time question/answer session with an interactive character;

A related object of the present invention is to implement such interactive character with an articulated response capability so that the user experiences a human-like interaction;

Still a further object of the present invention is to provide an INTERNET website with speech processing capability so that voice based data and commands can be used to interact with such site, thus enabling voice-based e-commerce and e-support services to be easily scaleable;

Another object is to implement a distributed speech recognition system that utilizes environmental variables as part of the recognition process to improve accuracy and speed;

A further object is to provide a scaleable query/response database system, to support any number of query topics and users as needed for a particular application and instantaneous demand;

Yet another object of the present invention is to provide a query recognition system that employs a two-step approach, including a relatively rapid first step to narrow down the list of potential responses to a smaller candidate set, and a second more computationally intensive second step to identify the best choice to be returned in response to the query from the candidate set;

A further object of the present invention is to provide a natural language processing system that facilitates query recognition by extracting lexical components of speech utterances, which components can be used for rapidly identifying a candidate set of potential responses appropriate for such speech utterances;

Another related object of the present invention is to provide a natural language processing system that facilitates query recognition by comparing lexical components of speech utterances with a candidate set of potential response to provide an extremely accurate best response to such query.

One general aspect of the present invention, therefore, relates to a natural language query system (NLQS) that offers a fully interactive method for answering user's questions over a distributed network such as the INTERNET or a local intranet. This interactive system when implemented over the worldwide web (WWW) services of the INTERNET functions so that a client or user can ask a question in a natural language such as English, French, German or Spanish and receive the appropriate answer at his or her personal computer also in his or her native natural language.

The system is distributed and consists of a set of integrated software modules at the client's machine and another set of integrated software programs resident on a server or set of servers. The client-side software program is comprised of a speech recognition program, an agent and its control program, and a communication program. The server-side program is comprised of a communication program, a natural language engine (NLE), a database processor (DBProcess), an interface program for interfacing the DBProcess with the NLE, and a SQL database. In addition, the client's machine is equipped with a microphone and a speaker. Processing of the speech utterance is divided between the client and server side so as to optimize processing and transmission latencies, and so as to provide support for even very thin client platforms.

In the context of an interactive learning application, the system is specifically used to provide a single-best answer to a user's question. The question that is asked at the client's machine is articulated by the speaker and captured by a microphone that is built in as in the case of a notebook computer or is supplied as a standard peripheral attachment. Once the question is captured, the question is processed partially by NLQS client-side software resident in the client's machine. The output of this partial processing is a set of speech vectors that are transported to the server via the INTERNET to complete the recognition of the user's questions. This recognized speech is then converted to text at the server.

After the user's question is decoded by the speech recognition engine (SRE) located at the server, the question is converted to a structured query language (SQL) query. This query is then simultaneously presented to a software process within the server called DBProcess for preliminary processing and to a Natural Language Engine (NLE) module for extracting the noun phrases (NP) of the user's question. During the process of extracting the noun phrase within the NLE, the tokens of the users' question are tagged. The tagged tokens are then grouped so that the NP list can be determined. This information is stored and sent to the DBProcess process.

In the DBProcess, the SQL query is fully customized using the NP extracted from the user's question and other environment variables that are relevant to the application. For example, in a training application, the user's selection of course, chapter and or section would constitute the environment variables. The SQL query is constructed using the extended SQL Full-Text predicates—CONTAINS, FREETEXT, NEAR, AND. The SQL query is next sent to the Full-Text search engine within the SQL database, where a Full-Text search procedure is initiated. The result of this search procedure is recordset of answers. This recordset contains stored questions that are similar linguistically to the user's question. Each of these stored questions has a paired answer stored in a separate text file, whose path is stored in a table of the database.

The entire recordset of returned stored answers is then returned to the NLE engine in the form of an array. Each stored question of the array is then linguistically processed sequentially one by one. This linguistic processing constitutes the second step of a 2-step algorithm to determine the single best answer to the user's question. This second step proceeds as follows: for each stored question that is returned in the recordset, a NP of the stored question is compared with the NP of the user's question. After all stored questions of the array are compared with the user's question, the stored question that yields the maximum match with the user's question is selected as the best possible stored question that matches the user's question. The metric that is used to determine the best possible stored question is the number of noun phrases.

The stored answer that is paired to the best-stored question is selected as the one that answers the user's question. The ID tag of the question is then passed to the DBProcess. This DBProcess returns the answer which is stored in a file.

A communication link is again established to send the answer back to the client in compressed form. The answer once received by the client is decompressed and articulated to the user by the text-to-speech engine. Thus, the invention can be used in any number of different applications involving interactive learning systems, INTERNET related commerce sites, INTERNET search engines, etc.

Computer-assisted instruction environments often require the assistance of mentors or live teachers to answer questions from students. This assistance often takes the form of organizing a separate pre-arranged forum or meeting time that is set aside for chat sessions or live call-in sessions so that at a scheduled time answers to questions may be provided. Because of the time immediacy and the on-demand or asynchronous nature of on-line training where a student may log on and take instruction at any time and at any location, it is important that answers to questions be provided in a timely and cost-effective manner so that the user or student can derive the maximum benefit from the material presented.

This invention addresses the above issues. It provides the user or student with answers to questions that are normally channeled to a live teacher or mentor. This invention provides a single-best answer to questions asked by the student. The student asks the question in his or her own voice in the language of choice. The speech is recognized and the answer to the question is found using a number of technologies including distributed speech recognition, full-text search database processing, natural language processing and text-to-speech technologies. The answer is presented to the user, as in the case of a live teacher, in an articulated manner by an agent that mimics the mentor or teacher, and in the language of choice—English, French, German, Japanese or other natural spoken language. The user can choose the agent's gender as well as several speech parameters such as pitch, volume and speed of the character's voice.

Other applications that benefit from NLQS are e-commerce applications. In this application, the user's query for a price of a book, compact disk or for the availability of any item that is to be purchased can be retrieved without the need to pick through various lists on successive web pages. Instead, the answer is provided directly to the user without any additional user input.

Similarly, it is envisioned that this system can be used to provide answers to frequently-asked questions (FAQs), and as a diagnostic service tool for e-support. These questions are typical of a give web site and are provided to help the user find information related to a payment procedure or the specifications of, or problems experienced with a product/service. In all of these applications, the NLQS architecture can be applied.

A number of inventive methods associated with these architectures are also beneficially used in a variety of INTERNET related applications.

Although the inventions are described below in a set of preferred embodiments, it will be apparent to those skilled in the art the present inventions could be beneficially used in many environments where it is necessary to implement fast, accurate speech recognition, and/or to provide a human-like dialog capability to an intelligent system.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram of a preferred embodiment of a natural language query system (NLQS) of the present invention, which is distributed across a client/server computing architecture, and can be used as an interactive learning system, an e-commerce system, an c-support system, and the like;

FIGS. **2**A–**2**C is a block diagram of a preferred embodiment of a client side system, including speech capturing modules, partial speech processing modules, encoding modules, transmission modules, agent control modules, and answer/voice feedback modules that can be used in the aforementioned NLQS;

FIG. **2**D is a block diagram of a preferred embodiment of a set of initialization routines and procedures used for the client side system of FIGS. **2**A–**2**C;

FIG. **3** is a block diagram of a preferred embodiment of a set of routines and procedures used for handling an iterated set of speech utterances on the client side system of FIG. **2**, transmitting speech data for such utterances to a remote server, and receiving appropriate responses back from such server;

FIG. **4** is a block diagram of a preferred embodiment of a set of initialization routines and procedures used for un-initializing the client side system of FIGS. **2**A–**2**C;

FIG. **4**A is a block diagram of a preferred embodiment of a set of routines and procedures used for implementing a distributed component of a speech recognition module for the server side system of FIG. **5**;

FIG. **4**B is a block diagram of a preferred set of routines and procedures used for implementing an SQL query builder for the server side system of FIG. **5**;

FIG. **4**C is a block diagram of a preferred embodiment of a set of routines and procedures used for implementing a database control process module for the server side system of FIG. **5**;

FIG. **4**D is a block diagram of a preferred embodiment of a set of routines and procedures used for implementing a natural language engine that provides query formulation support, a query response module, and an interface to the database control process module for the server side system of FIG. **5**;

FIG. **5** is a block diagram of a preferred embodiment of a server side system, including a speech recognition module to complete processing of the speech utterances, environmental and grammar control modules, query formulation modules, a natural language engine, a database control module, and a query response module that can be used in the aforementioned NLQS;

FIG. **6** illustrates the organization of a full-text database used as part of server side system shown in FIG. **5**;

FIG. **7**A illustrates the organization of a full-text database course table used as part of server side system shown in FIG. **5** for an interactive learning embodiment of the present invention;

FIG. **7**B illustrates the organization of a full-text database chapter table used as part of server side system shown in FIG. **5** for an interactive learning embodiment of the present invention;

FIG. **7**C describes the fields used in a chapter table used as part of server side system shown in FIG. **5** for an interactive learning embodiment of the present invention;

FIG. **7**D describes the fields used in a section table used as part of server side system shown in FIG. **5** for an interactive learning embodiment of the present invention;

FIG. **8** is a flow diagram of a first set of operations performed by a preferred embodiment of a natural language engine on a speech utterance including Tokenization, Tagging and Grouping;

FIG. **9** is a flow diagram of the operations performed by a preferred embodiment of a natural language engine on a speech utterance including stemming and Lexical Analysis

FIG. **10** is a block diagram of a preferred embodiment of a SQL database search and support system for the present invention;

FIGS. **11**A–**11**C are flow diagrams illustrating steps performed in a preferred two step process implemented for query recognition by the NLQS of FIG. **2**;

FIG. **12** is an illustration of another embodiment of the present invention implemented as part of a Web-based speech based learning/training System;

FIGS. **13**–**17** are illustrations of another embodiment of the present invention implemented as part of a Web-based e-commerce system;

FIG. **18** is an illustration of another embodiment of the present invention implemented as part of a voice-based Help Page for an E-Commerce Web Site.

DETAILED DESCRIPTION OF THE INVENTION

Overview

As alluded to above, the present inventions allow a user to ask a question in a natural language such as English, French, German, Spanish or Japanese at a client computing system (which can be as simple as a personal digital assistant or cell-phone, or as sophisticated as a high end desktop PC) and receive an appropriate answer from a remote server also in his or her native natural language. As such, the embodiment of the invention shown in FIG. **1** is beneficially used in what can be generally described as a Natural Language Query System (NLQS) **100**, which is configured to interact on a real-time basis to give a human-like dialog capability/experience for e-commerce, e-support, and e-learning applications.

The processing for NLQS **100** is generally distributed across a client side system **150**, a data link **160**, and a server-side system **180**. These components are well known in the art, and in a preferred embodiment include a personal computer system **150**, an INTERNET connection **160**A, **160**B, and a larger scale computing system **180**. It will be understood by those skilled in the art that these are merely exemplary components, and that the present invention is by no means limited to any particular implementation or combination of such systems. For example, client-side system **150** could also be implemented as a computer peripheral, a PDA, as part of a cell-phone, as part of an INTERNET-adapted appliance, an INTERNET linked kiosk, etc. Similarly, while an INTERNET connection is depicted for data link **160**A, it is apparent that any channel that is suitable for carrying data between client system **150** and server system **180** will suffice, including a wireless link, an RF link, an IR link, a LAN, and the like. Finally, it will be further appreciated that server system **180** may be a single, large-scale system, or a collection of smaller systems interlinked to support a number of potential network users.

Initially speech input is provided in the form of a question or query articulated by the speaker at the client's machine or personal accessory as a speech utterance. This speech utterance is captured and partially processed by NLQS client-side software **155** resident in the client's machine. To facilitate and enhance the human-like aspects of the

interaction, the question is presented in the presence of an animated character **157** visible to the user who assists the user as a personal information retriever/agent. The agent can also interact with the user using both visible text output on a monitor/display (not shown) and/or in audible form using a text to speech engine **159**. The output of the partial processing done by SRE **155** is a set of speech vectors that are transmitted over communication channel **160** that links the user's machine or personal accessory to a server or servers via the INTERNET or a wireless gateway that is linked to the INTERNET as explained above. At server **180**, the partially processed speech signal data is handled by a server-side SRE **182**, which then outputs recognized speech text corresponding to the user's question. Based on this user question related text, a text-to-query converter **184** formulates a suitable query that is used as input to a database processor **186**. Based on the query, database processor **186** then locates and retrieves an appropriate answer using a customized SQL query from database **188**. A Natural Language Engine **190** facilitates structuring the query to database **188**. After a matching answer to the user's question is found, the former is transmitted in text form across data link **160**B, where it is converted into speech by text to speech engine **159**, and thus expressed as oral feedback by animated character agent **157**.

Because the speech processing is broken up in this fashion, it is possible to achieve real-time, interactive, human-like dialog consisting of a large, controllable set of questions/answers. The assistance of the animated agent **157** further enhances the experience, making it more natural and comfortable for even novice users. To make the speech recognition process more reliable, context-specific grammars and dictionaries are used, as well as natural language processing routines at NLE **190**, to analyze user questions lexically. While context-specific processing of speech data is known in the art (see e.g., U.S. Pat. Nos. 5,960,394, 5,867, 817, 5,758,322 and 5,384,892 incorporated by reference herein) the present inventors are unaware of any such implementation as embodied in the present inventions. The text of the user's question is compared against text of other questions to identify the question posed by the user by DB processor/engine (DBE) **186**. By optimizing the interaction and relationship of the SR engines **155** and **182**, the NLP routines **190**, and the dictionaries and grammars, an extremely fast and accurate match can be made, so that a unique and responsive answer can be provided to the user.

On the server side **180**, interleaved processing further accelerates the speech recognition process. In simplified terms, the query is presented simultaneously both to NLE **190** after the query is formulated, as well as to DBE **186**. NLE **190** and SRE **182** perform complementary functions in the overall recognition process. In general, SRE **182** is primarily responsible for determining the identity of the words articulated by the user, while NLE **190** is responsible for the linguistic morphological analysis of both the user's query and the search results returned after the database query.

After the user's query is analyzed by NLE **190** some parameters are extracted and sent to the DBProcess. Additional statistics are stored in an array for the 2^{nd} step of processing. During the 2nd step of 2-step algorithm, the recordset of preliminary search results are sent to the NLE **160** for processing. At the end of this 2nd step, the single question that matches the user's query is sent to the DBProcess where further processing yields the paired answer that is paired with the single best stored question.

Thus, the present invention uses a form of natural language processing (NLP) to achieve optimal performance in a speech based web application system. While NLP is known in the art, prior efforts in Natural Language Processing (NLP) work nonetheless have not been well integrated with Speech Recognition (SR) technologies to achieve reasonable results in a web-based application environment. In speech recognition, the result is typically a lattice of possible recognized words each with some probability of fit with the speech recognizer. As described before, the input to a typical NLP system is typically a large linguistic unit. The NLP system is then charged with the parsing, understanding and indexing of this large linguistic unit or set of transcribed utterances. The result of this NLP process is to understand lexically or morphologically the entire linguistic unit as opposed to word recognition. Put another way, the linguistic unit or sentence of connected words output by the SRE has to be understood lexically, as opposed to just being "recognized".

As indicated earlier, although speech recognition technology has been available for several years, the technical requirements for the NLQS invention have placed severe restrictions on the specifications for the speech recognition accuracy that is required for an application that combines speech recognition and natural language processing to work satisfactorily. In realizing that even with the best of conditions, it might be not be possible to achieve the perfect 100% speech recognition accuracy that is required, the present invention employs an algorithm that balances the potential risk of the speech recognition process with the requirements of the natural language processing so that even in cases where perfect speech recognition accuracy is not achieved for each word in the query, the entire query itself is nonetheless recognized with sufficient accuracy.

This recognition accuracy is achieved even while meeting very stringent user constraints, such as short latency periods of 3 to 5 seconds (ideally—ignoring transmission latencies which can vary) for responding to a speech-based query, and for a potential set of 100–250 query questions. This quick response time gives the overall appearance and experience of a real-time discourse that is more natural and pleasant from the user's perspective. Of course, non-real time applications, such as translation services for example, can also benefit from the present teachings as well, since a centralized set of HMMs, grammars, dictionaries, etc., are maintained.

General Aspects of Speech Recognition Used in the Present Inventions

General background information on speech recognition can be found in the prior art references discussed above and incorporated by reference herein. Nonetheless, a discussion of some particular exemplary forms of speech recognition structures and techniques that are well-suited for NLQS **100** is provided next to better illustrate some of the characteristics, qualities and features of the present inventions.

Speech recognition technology is typically of two types—speaker independent and speaker dependent. In speaker-dependent speech recognition technology, each user has a voice file in which a sample of each potentially recognized word is stored. Speaker-dependent speech recognition systems typically have large vocabularies and dictionaries making them suitable for applications as dictation and text transcribing. It follows also that the memory and processor resource requirements for the speaker-dependent can be and are typically large and intensive.

Conversely speaker-independent speech recognition technology allows a large group of users to use a single vocabulary file. It follows then that the degree of accuracy that can

be achieved is a function of the size and complexity of the grammars and dictionaries that can be supported for a given language. Given the context of applications for which NLQS, the use of small grammars and dictionaries allow speaker independent speech recognition technology to be implemented in NLQS.

The key issues or requirements for either type—speaker-independent or speaker-dependent, are accuracy and speed. As the size of the user dictionaries increase, the speech recognition accuracy metric—word error rate (WER) and the speed of recognition decreases. This is so because the search time increases and the pronunciation match becomes more complex as the size of the dictionary increases.

The basis of the NLQS speech recognition system is a series of Hidden Markov Models (HMM), which, as alluded to earlier, are mathematical models used to characterize any time varying signal. Because parts of speech are considered to be based on an underlying sequence of one or more symbols, the HMM models corresponding to each symbol are trained on vectors from the speech waveforms. The Hidden Markov Model is a finite set of states, each of which is associated with a (generally multi-dimensional) probability distribution. Transitions among the states are governed by a set of probabilities called transition probabilities. In a particular state an outcome or observation can be generated, according to an associated probability distribution. This finite state machine changes state once every time unit, and each time t such that a state j is entered, a spectral parameter vector O_t is generated with probability density $B_j(O_t)$. It is only the outcome, not the state which is visible to an external observer and therefore states are "hidden" to the outside; hence the name Hidden Markov Model.

In isolated speech recognition, it is assumed that the sequence of observed speech vectors corresponding to each word can each be described by a Markov model as follows:

$$O=o_1, o_2, \ldots o_T \tag{1-1}$$

where o_t is a speech vector observed at time t. The isolated word recognition then is to compute:

$$\arg\max\{P(w_i|O)\} \tag{1-2}$$

By using Bayes' Rule,

$$\{P(w_i|O)\}=[P(O|w_i)P(w_i)]/P(O) \tag{1-3}$$

In the general case, the Markov model when applied to speech also assumes a finite state machine which changes state once every time unit and each time that a state j is entered, a speech vector o_t is generated from the probability density $b_j(o_t)$. Furthermore, the transition from state i to state j is also probabilistic and is governed by the discrete probability a_{ij}.

For a state sequence X, the joint probability that 0 is generated by the model M moving through a state sequence X is the product of the transition probabilities and the output probabilities. Only the observation sequence is known—the state sequence is hidden as mentioned before.

Given that X is unknown, the required likelihood is computed by summing over all possible state sequences X=x(1), x(2), x(3), . . . x(T), that is

$$P(O|M)=\Sigma\{a_{x(0)\,x(1)}\Pi b(x)\,(o_t)a_{x(t)\,x(t+1)}\}$$

Given a set of models M_i, corresponding to words w_i equation 1-2 is solved by using 1-3 and also by assuming that:

$$P(O|w_i)=P(O|M_i)$$

All of this assumes that the parameters $\{a_{ij}\}$ and $\{b_j(o_t)\}$ are known for each model M_i. This can be done, as explained earlier, by using a set of training examples corresponding to a particular model. Thereafter, the parameters of that model can be determined automatically by a robust and efficient re-estimation procedure. So if a sufficient number of representative examples of each word are collected, then a HMM can be constructed which simply models all of the many sources of variability inherent in real speech. This training is well-known in the art, so it is not described at length herein, except to note that the distributed architecture of the present invention enhances the quality of HMMs, since they are derived and constituted at the server side, rather than the client side. In this way, appropriate samples from users of different geographical areas can be easily compiled and analyzed to optimize the possible variations expected to be seen across a particular language to be recognized. Uniformity of the speech recognition process is also well-maintained, and error diagnostics are simplified, since each prospective user is using the same set of HMMs during the recognition process.

To determine the parameters of a HMM from a set of training samples, the first step typically is to make a rough guess as to what they might be. Then a refinement is done using the Baum-Welch estimation formulae. By these formulae, the maximum likelihood estimates of μ_j (where μ_j is mean vector and Σ_j is covariance matrix) is:

$$\mu_j=\Sigma^T_{t=1}L_j(t)o_t/[\Sigma^T_{t=1}L_j(t)o_t]$$

A forward-backward algorithm is next used to calculate the probability of state occupation $L_j(t)$. If the forward probability $\alpha_j(t)$ for some model M with N states is defined as:

$$\alpha_j(t)=P(o_1, \ldots, o_t, x(t)=j|M)$$

This probability can be calculated using the recursion:

$$\alpha_j(t)=[\Sigma^{N-1}_{i=2}\alpha(t-1)a_{ij}]b_j(o_t)$$

Similarly the backward probability can be computed using the recursion:

$$\beta_j(t)=\Sigma^{N-1}_{j=2}\,a_{ij}b_j(o_{t+1})(t+1)$$

Realizing that the forward probability is a joint probability and the backward probability is a conditional probability, the probability of state occupation is the product of the two probabilities:

$$\alpha j(t)\beta_j(t)=P(O,x(t)=j|M)$$

Hence the probability of being in state j at a time t is:

$$L_j(t)=1/P[\alpha_j(t)\beta_j(t)]$$

where P=P(O|M)

To generalize the above for continuous speech recognition, we assume the maximum likelihood state sequence where the summation is replaced by a maximum operation. Thus for a given model M, let ϕj (t) represent the maximum likelihood of observing speech vectors o_1 to o_t and being used in state j at time t:

$$\phi_j(t)=\max\{\phi j(t)(t-1)\alpha_{ij}\}\beta_j(o_t)$$

Expressing this logarithmically to avoid underflow, this likelihood becomes:

$$\psi_j(t)=\max\{\psi_i(t-1)+\log(\alpha_{ij})\}+\log(b_j(o_t)$$

This is also known as the Viterbi algorithm. It can be visualized as finding the best path through a matrix where the vertical dimension represents the states of the HMM and horizontal dimension represents frames of speech i.e. time. To complete the extension to connected speech recognition, it is further assumed that each HMM representing the underlying sequence is connected. Thus the training data for continuous speech recognition should consist of connected utterances; however, the boundaries between words do not have to be known.

To improve computational speed/efficiency, the Viterbi algorithm is sometimes extended to achieve convergence by using what is known as a Token Passing Model. The token passing model represents a partial match between the observation sequence o_1 to o_t and a particular model, subject to the constraint that the model is in state j at time t. This token passing model can be extended easily to connected speech environments as well if we allow the sequence of HMMs to be defined as a finite state network. A composite network that includes both phoneme-based HMMs and complete words can be constructed so that a single-best word can be recognized to form connected speech using word N-best extraction from the lattice of possibilities. This composite form of HMM-based connected speech recognizer is the basis of the NLQS speech recognizer module. Nonetheless, the present invention is not limited as such to such specific forms of speech recognizers, and can employ other techniques for speech recognition if they are otherwise compatible with the present architecture and meet necessary performance criteria for accuracy and speed to provide a real-time dialog experience for users.

The representation of speech for the present invention's HMM-based speech recognition system assumes that speech is essentially either a quasi-periodic pulse train (for voiced speech sounds) or a random noise source (for unvoiced sounds). It may be modeled as two sources—one a impulse train generator with pitch period P and a random noise generator which is controlled by a voice/unvoiced switch. The output of the switch is then fed into a gain function estimated from the speech signal and scaled to feed a digital filter H(z) controlled by the vocal tract parameter characteristics of the speech being produced. All of the parameters for this model—the voiced/unvoiced switching, the pitch period for voiced sounds, the gain parameter for the speech signal and the coefficient of the digital filter, vary slowly with time. In extracting the acoustic parameters from the user's speech input so that it can evaluated in light of a set of HMMs, cepstral analysis is typically used to separate the vocal tract information from the excitation information. The cepstrum of a signal is computed by taking the Fourier (or similar) transform of the log spectrum. The principal advantage of extracting cepstral coefficients is that they are de-correlated and the diagonal covariances can be used with HMMs. Since the human ear resolves frequencies non-linearly across the audio spectrum, it has been shown that a front-end that operates in a similar non-linear way improves speech recognition performance.

Accordingly, instead of a typical linear prediction-based analysis, the front-end of the NLQS speech recognition engine implements a simple, fast Fourier transform based filter bank designed to give approximately equal resolution on the Mel-scale. To implement this filter bank, a window of speech data (for a particular time frame) is transformed using a software based Fourier transform and the magnitude taken. Each FFT magnitude is then multiplied by the corresponding filter gain and the results accumulated. The cepstral coefficients that are derived from this filter-bank analysis at the front end are calculated during a first partial processing phase of the speech signal by using a Discrete Cosine Transform of the log filter bank amplitudes. These cepstral coefficients are called Mel-Frequency Cepstral Coefficients (MFCC) and they represent some of the speech parameters transferred from the client side to characterize the acoustic features of the user's speech signal. These parameters are chosen for a number of reasons, including the fact that they can be quickly and consistently derived even across systems of disparate capabilities (i.e., for everything from a low power PDA to a high powered desktop system), they give good discrimination, they lend themselves to a number of useful recognition related manipulations, and they are relatively small and compact in size so that they can be transported rapidly across even a relatively narrow band link. Thus, these parameters represent the least amount of information that can be used by a subsequent server side system to adequately and quickly complete the recognition process.

To augment the speech parameters an energy term in the form of the logarithm of the signal energy is added. Accordingly, RMS energy is added to the 12 MFCC's to make 13 coefficients. These coefficients together make up the partially processed speech data transmitted in compressed form from the user's client system to the remote server side.

The performance of the present speech recognition system is enhanced significantly by computing and adding time derivatives to the basic static MFCC parameters at the server side. These two other sets of coefficients—the delta and acceleration coefficients representing change in each of the 13 values from frame to frame (actually measured across several frames), are computed during a second partial speech signal processing phase to complete the initial processing of the speech signal, and are added to the original set of coefficients after the latter are received. These MFCCs together with the delta and acceleration coefficients constitute the observation vector O_t mentioned above that is used for determining the appropriate HMM for the speech data.

The delta and acceleration coefficients are computed using the following regression formula:

$$d_t = \Sigma^{\theta}_{\theta=1}[c_{t+\theta} - c_{t-\theta}]/2\Sigma^{\theta}_{\theta=1}\theta^2$$

where d_t is a delta coefficient at time t computed in terms of the corresponding static coefficients:

$$d_t = [c_{t+\theta} - c_{t-\theta}]/2\theta$$

In a typical stand-alone implementation of a speech recognition system, the entire SR engine runs on a single client. In other words, both the first and second partial processing phases above are executed by the same DSP (or microprocessor) running a ROM or software code routine at the client's computing machine.

In contrast, because of several considerations, specifically—cost, technical performance, and client hardware uniformity, the present NLQS system uses a partitioned or distributed approach. While some processing occurs on the client side, the main speech recognition engine runs on a centrally located server or number of servers. More specifically, as noted earlier, capture of the speech signals, MFCC vector extraction and compression are implemented on the client's machine during a first partial processing phase. The routine is thus streamlined and simple enough to be implemented within a browser program (as a plug in module, or a downloadable applet for example) for maximum ease of use and utility. Accordingly, even very "thin"

client platforms can be supported, which enables the use of the present system across a greater number of potential sites. The primary MFCCs are then transmitted to the server over the channel, which, for example, can include a dial-up INTERNET connection, a LAN connection, a wireless connection and the like. After decompression, the delta and acceleration coefficients are computed at the server to complete the initial speech processing phase, and the resulting observation vectors O_t are also determined.

General Aspects of Speech Recognition Engine

The speech recognition engine is also located on the server, and is based on a HTK-based recognition network compiled from a word-level network, a dictionary and a set of HMMs. The recognition network consists of a set of nodes connected by arcs. Each node is either a HMM model instance or a word end. Each model node is itself a network consisting of states connected by arcs. Thus when fully compiled, a speech recognition network consists of HMM states connected by transitions. For an unknown input utterance with T frames, every path from the start node to the exit node of the network passes through T HMM states. Each of these paths has log probability which is computed by summing the log probability of each individual transition in the path and the log probability of each emitting state generating the corresponding observation. The function of the Viterbi decoder is find those paths through the network which have the highest log probability. This is found using the Token Passing algorithm. In a network that has many nodes, the computation time is reduced by only allowing propagation of those tokens which will have some chance of becoming winners. This process is called pruning.

Natural Language Processor

In a typical natural language interface to a database, the user enters a question in his/her natural language, for example, English. The system parses it and translates it to a query language expression. The system then uses the query language expression to process the query and if the search is successful, a recordset representing the results is displayed in English either formatted as raw text or in a graphical form. For a natural language interface to work well involves a number of technical requirements.

For example, it needs to be robust—in the sentence 'What's the departments turnover' it needs to decide that the word whats=what's=what is. And it also has to determine that departments=department's. In addition to being robust, the natural language interface has to distinguish between the several possible forms of ambiguity that may exist in the natural language—lexical, structural, reference and ellipsis ambiguity. All of these requirements, in addition to the general ability to perform basic linguistic morphological operations of tokenization, tagging and grouping, are implemented within the present invention.

Tokenization is implemented by a text analyzer which treats the text as a series of tokens or useful meaningful units that are larger than individual characters, but smaller than phrases and sentences. These include words, separable parts of words, and punctuation. Each token is associated with an offset and a length. The first phase of tokenization is the process of segmentation which extracts the individual tokens from the input text and keeps track of the offset where each token originated in the input text. The tokenizer output lists the offset and category for each token. In the next phase of the text analysis, the tagger uses a built-in morphological analyzer to look up each word/token in a phrase or sentence and internally lists all parts of speech. The output is the input string with each token tagged with a parts of speech notation. Finally the grouper which functions as a phrase extractor or phrase analyzer, determines which groups of words form phrases. These three operations which are the foundations for any modem linguistic processing schemes, are fully implemented in optimized algorithms for determining the single-best possible answer to the user's question.

SQL Database and Full-Text Query

Another key component of present system is a SQL-database. This database is used to store text, specifically the answer-question pairs are stored in full-text tables of the database. Additionally, the full-text search capability of the database allows full-text searches to be carried out.

While a large portion of all digitally stored information is in the form of unstructured data, primarily text, it is now possible to store this textual data in traditional database systems in character-based columns such as varchar and text. In order to effectively retrieve textual data from the database, techniques have to be implemented to issue queries against textual data and to retrieve the answers in a meaningful way where it provides the answers as in the case of the NLQS system.

There are two major types of textual searches: Property—This search technology first applies filters to documents in order to extract properties such as author, subject, type, word count, printed page count, and time last written, and then issues searches against those properties; Full-text—this search technology first creates indexes of all non-noise words in the documents, and then uses these indexes to support linguistic searches and proximity searches.

Two additional technologies are also implemented in this particular RDBMs: SQL Server also have been integrated: A Search service—a full-text indexing and search service that is called both index engine and search, and a parser that accepts full-text SQL extensions and maps them into a form that can be processed by the search engine.

The four major aspects involved in implementing full-text retrieval of plain-text data from a full-text-capable database are: Managing the definition of the tables and columns that are registered for full-text searches; Indexing the data in registered columns—the indexing process scans the character streams, determines the word boundaries (this is called word breaking), removes all noise words (this also is called stop words), and then populates a full-text index with the remaining words; Issuing queries against registered columns for populated full-text indexes; Ensuring that subsequent changes to the data in registered columns gets propagated to the index engine to keep the full-text indexes synchronized.

The underlying design principle for the indexing, querying, and synchronizing processes is the presence of a full-text unique key column (or single-column primary key) on all tables registered for full-text searches. The full-text index contains an entry for the non-noise words in each row together with the value of the key column for each row.

When processing a full-text search, the search engine returns to the database the key values of the rows that match the search criteria.

The full-text administration process starts by designating a table and its columns of interest for full-text search. Customized NLQS stored procedures are used first to register tables and columns as eligible for full-text search. After that, a separate request by means of a stored procedure is issued to populate the full-text indexes. The result is that the underlying index engine gets invoked and asynchronous index population begins. Full-text indexing tracks which significant words are used and where they are located. For example, a full-text index might indicate that the word "NLQS" is found at word number **423** and word number **982** in the Abstract column of the DevTools table for the row

associated with a ProductID of 6. This index structure supports an efficient search for all items containing indexed words as well as advanced search operations, such as phrase searches and proximity searches. (An example of a phrase search is looking for "white elephant," where "while" is followed by "elephant". An example of a proximity search is looking for "big" and "house" where "big" occurs near "house".) To prevent the full-text index from becoming bloated, noise words such as "a," "and," and "the" are ignored.

Extensions to the Transact-SQL language are used to construct full-text queries. The two key predicates that are used in the NLQS are CONTAINS and FREETEXT.

The CONTAINS predicate is used to determine whether or not values in full-text registered columns contain certain words and phrases. Specifically, this predicate is used to search for:

A word or phrase.

The prefix of a word or phrase.

A word or phrase that is near another.

A word that is an inflectional form of another (for example, "drive" is the inflectional stem of "drives," "drove," "driving" and "driven").

A set of words or phrases, each of which is assigned a different weighting.

The relational engine within SQL Server recognizes the CONTAINS and FREETEXT predicates and performs some minimal syntax and semantic checking, such as ensuring that the column referenced in the predicate has been registered for full-text searches. During query execution, a full-text predicate and other relevant information are passed to the full-text search component. After further syntax and semantic validation, the search engine is invoked and returns the set of unique key values identifying those rows in the table that satisfy the full-text search condition. In addition to the FREETEXT and CONTAINS, other predicates such as AND, LIKE, NEAR are combined to create the customized NLQS SQL construct.

Full-Text Query Architecture of the SQL Database

The full-text query architecture is comprised of the following several components—Full-Text Query component, the SQL Server Relational Engine, the Full-Text provider and the Search Engine.

The Full-Text Query component of the SQL database accept a full-text predicate or rowset-valued function from the SQL Server; transform parts of the predicate into an internal format, and sends it to Search Service, which returns the matches in a rowset. The rowset is then sent back to SQL Server. SQL Server uses this information to create the resultset that is then returned to the submitter of the query.

The SQL Server Relational Engine accepts the CONTAINS and FREETEXT predicates as well as the CONTAINSTABLE( ) and FREETEXTTABLE( ) rowset-valued functions. During parse time, this code checks for conditions such as attempting to query a column that has not been registered for full-text search. If valid, then at run time, the ft_search_condition and context information is sent to the full-text provider. Eventually, the full-text provider returns a rowset to SQL Server, which is used in any joins (specified or implied) in the original query. The Full-Text Provider parses and validates the ft_search_condition, constructs the appropriate internal representation of the full-text search condition, and then passes it to the search engine. The result is returned to the relational engine by means of a rowset of rows that satisfy ft_search_condition.

Client Side System **150**

The architecture of client-side system **150** of Natural Language Query System **100** is illustrated in greater detail in FIGS. **2**A–**2**C. Referring to FIGS. **2**A–**2**C, the three main processes effectuated by Client System **150** are illustrated as follows: Initialization process **200**A consisting of SRE **201**, Communication **202** and Microsoft (MS) Agent **203** routines; At FIG. **2**B on an iterative process **200**B consisting of two sub-routines: a) Receive User Speech **208**—made up of SRE **204** and Communication **205**; and b) Receive Answer from Server **207**—made up of MS Speak Agent **206**, Communication **209**, Voice data file **210** and Text to Speech Engine **211**. Finally, in FIG. **2**C un-initialization process **200**C is made up of three sub-routines: SRE **212**, Communication **213**, and MS Agent **214**. Each of the above three processes are described in detail in the following paragraphs. It will be appreciated by those skilled in the art that the particular implementation for such processes and routines will vary from client platform to platform, so that in some environments such processes may be embodied in hard-coded routines executed by a dedicated DSP, while in others they may be embodied as software routines executed by a shared host processor, and in still others a combination of the two may be used.

Initialization at Client System **150**

The initialization of the Client System **150** is illustrated in FIG. **2**D and is comprised generally of 3 separate initializing processes: client-side Speech Recognition Engine **220**A, MS Agent **220**B and Communication processes **220**C.

Initialization of Speech Recognition Engine **220**A

Speech Recognition Engine **155** is initialized and configured using the routines shown in **220**A. First, an SRE COM Library is initialized. Next, memory **220** is allocated to hold Source and Coder objects, are created by a routine **221**. Loading of configuration file **221**A from configuration data file **221**B also takes place at the same time that the SRE Library is initialized. In configuration file **221**B, the type of the input of Coder and the type of the output of the Coder are declared. The structure, operation, etc. of such routines are well-known in the art, and they can be implemented using a number of fairly straightforward approaches. Accordingly, they are not discussed in detail herein. Next, Speech and Silence components of an utterance are calibrated using a routine **222**, in a procedure that is also well-known in the art. To calibrate the speech and silence components, the user preferably articulates a sentence that is displayed in a text box on the screen. The SRE library then estimates the noise and other parameters required to find e silence and speech elements of future user utterances.

Initialization of MS Agent **220**B

The software code used to initialize and set up a MS Agent **220**B is also illustrated in FIG. **2**D. The MS Agent **220**B routine is responsible for coordinating and handling the actions of the animated agent **157** (FIG.**1**). This initialization thus consists of the following steps:

1. Initialize COM library **223**. This part of the code initializes the COM library, which is required to use ActiveX Controls, which controls are well-known in the art.
2. Create instance of Agent Server **224**—this part of the code creates an instance of Agent ActiveX control.
3. Loading of MS Agent **225**—this part of the code loads MS Agent character from a specified file **225**A containing general parameter data for the Agent Character, such as the overall appearance, shape, size, etc.
4. Get Character Interface **226**—this part of the code gets an appropriate interface for the specified character; for example, characters may have different control/interaction capabilities that can be presented to the user.

5. Add Commands to Agent Character Option **227**—this part of the code adds commands to an Agent Properties sheet, which sheet can be accessed by clicking on the icon that appears in the system tray, when the Agent character is loaded e.g., that the character can Speak, how he/she moves, TTS Properties, etc.
6. Show the Agent Character **228**—this part of the code displays the Agent character on the screen so it can be seen by the user;
7. AgentNotifySink—to handle events. This part of the code creates AgentNotifySink object **229**, registers it at **230** and then gets the Agent Properties interface **231**. The property sheet for the Agent character is assigned using routine **232**.
8. Do Character Animations **233**—This part of the code plays specified character animations to welcome the user to NLQS **100**.

The above then constitutes the entire sequence required to initialize the MS Agent. As with the SRE routines, the MS Agent routines can be implemented in any suitable and conventional fashion by those skilled in the art based on the present teachings. The particular structure, operation, etc. of such routines is not critical, and thus they are not discussed in detail herein.

In a preferred embodiment, the MS Agent is configured to have an appearance and capabilities that are appropriate for the particular application. For instance, in a remote learning application, the agent has the visual form and mannerisms/attitude/gestures of a college professor. Other visual props (blackboard, textbook, etc.) may be used by the agent and presented to the user to bring to mind the experience of being in an actual educational environment. The characteristics of the agent may be configured at the client side **150**, and/or as part of code executed by a browser program (not shown) in response to configuration data and commands from a particular web page. For example, a particular website offering medical services may prefer to use a visual image of a doctor. These and many other variations will be apparent to those skilled in the art for enhancing the human-like, real-time dialog experience for users.

Initialization of Communication Link **160**A

The initialization of Communication Link **160**A is shown with reference to process **220**C FIG. **2**D. Referring to FIG. **2**D, this initialization consists of the following code components: Open INTERNET Connection **234**—this part of the code opens an INTERNET Connection and sets the parameter for the connection. Then Set Callback Status routine **235** sets the callback status so as to inform the user of the status of connection. Finally Start New HTTP INTERNET Session **236** starts a new INTERNET session. The details of Communications Link **160** and the set up process **220**C are not critical, and will vary from platform to platform. Again, in some cases, users may use a low-speed dial-up connection, a dedicated high speed switched connection (T1 for example), an always-on xDSL connection, a wireless connection, and the like.

Iterative Processing of Queries/Answers

As illustrated in FIG. **3**, once initialization is complete, an iterative query/answer process is launched when the user presses the Start Button to initiate a query. Referring to FIG. **3**, the iterative query/answer process consists of two main sub-processes implemented as routines on the client side system **150**: Receive User Speech **240** and Receive User Answer **243**. The Receive User Speech **240** routine receives speech from the user (or another audio input source), while the Receive User Answer **243** routine receives an answer to the user's question in the form of text from the server so that it can be converted to speech for the user by text-to-speech engine **159**. As used herein, the term "query" is referred to in the broadest sense to refer, to either a question, a command, or some form of input used as a control variable by the system. For example, a query may consist of a question directed to a particular topic, such as "what is a network" in the context of a remote learning application. In an e-commerce application a query might consist of a command to "list all books by Mark Twain" for example. Similarly, while the answer in a remote learning application consists of text that is rendered into audible form by the text to speech engine **159**, it could also be returned as another form of multi-media information, such as a graphic image, a sound file, a video file, etc. depending on the requirements of the particular application. Again, given the present teachings concerning the necessary structure, operation, functions, performance, etc., of the client-side Receive User Speech **240** and Receiver User Answer **243** routines, one of ordinary skill in the art could implement such in a variety of ways.

Receive User Speech—As illustrated in FIG. **3**, the Receive User Speech routine **240** consists of a SRE **241** and a Communication **242** process, both implemented again as routines on the client side system **150** for receiving and partially processing the user's utterance. SRE routine **241** uses a coder **248** which is prepared so that a coder object receives speech data from a source object. Next the Start Source **249** routine is initiated. This part of the code initiates data retrieval using the source Object which will in turn be given to the Coder object. Next, MFCC vectors **250** are extracted from the Speech utterance continuously until silence is detected. As alluded to earlier, this represents the first phase of processing of the input speech signal, and in a preferred embodiment, it is intentionally restricted to merely computing the MFCC vectors for the reasons already expressed above. These vectors include the 12 cepstral coefficients and the RMS energy term, for a total of 13 separate numerical values for the partially processed speech signal.

In some environments, nonetheless, it is conceivable that the MFCC delta parameters and MFCC acceleration parameters can also be computed at client side system **150**, depending on the computation resources available, the transmission bandwidth in data link **160**A available to server side system **180**, the speed of a transceiver used for carrying data in the data link, etc. These parameters can be determined automatically by client side system upon initializing SRE **155** (using some type of calibration routine to measure resources), or by direct user control, so that the partitioning of signal processing responsibilities can be optimized on a case-by-case basis. In some applications, too, server side system **180** may lack the appropriate resources or routines for completing the processing of the speech input signal. Therefore, for some applications, the allocation of signal processing responsibilities may be partitioned differently, to the point where in fact both phases of the speech signal processing may take place at client side system **150** so that the speech signal is completely—rather than partially—processed and transmitted for conversion into a query at server side system **180**.

Again in a preferred embodiment, to ensure reasonable accuracy and real-time performance from a query/response perspective, sufficient resources are made available in a client side system so that 100 frames per second of speech data can be partially processed and transmitted through link **160**A. Since the least amount of information that is necessary to complete the speech recognition process (only 13

coefficients) is sent, the system achieves a real-time performance that is believed to be highly optimized, because other latencies (i.e., client-side computational latencies, packet formation latencies, transmission latencies) are minimized. It will be apparent that the principles of the present invention can be extended to other SR applications where some other methodology is used for breaking down the speech input signal by an SRE (i.e., non-MFCC based). The only criteria is that the SR processing be similarly dividable into multiple phases, and with the responsibility for different phases being handled on opposite sides of link **160**A depending on overall system performance goals, requirements and the like. This functionality of the present invention can thus be achieved on a system-by-system basis, with an expected and typical amount of optimization being necessary for each particular implementation.

Thus, the present invention achieves a response rate performance that is tailored in accordance with the amount of information that is computed, coded and transmitted by the client side system **150**. So in applications where real-time performance is most critical, the least possible amount of extracted speech data is transmitted to reduce these latencies, and, in other applications, the amount of extracted speech data that is processed, coded and transmitted can be varied.

Communication—transmit communication module **242** is used to implement the transport of data from the client to the server over the data link **160**A, which in a preferred embodiment is the INTERNET. As explained above, the data consists of encoded MFCC vectors that will be used at then server-side of the Speech Recognition engine to complete the speech recognition decoding. The sequence of the communication is as follows:

OpenHTTPRequest **251**—this part of the code first converts MFCC vectors to a stream of bytes, and then processes the bytes so that it is compatible with a protocol known as HTTP. This protocol is well-known in the art, and it is apparent that for other data links another suitable protocol would be used.

1. Encode MFCC Byte Stream **251**—this part of the code encodes the MFCC vectors, so that they can be sent to the server via HTTP.
2. Send data **252**—this part of the code sends MFCC vectors to the server using the INTERNET connection and the HTTP protocol.

Wait for the Server Response **253**—this part of the code monitors the data link **160**A a response from server side system **180** arrives. In summary, the MFCC parameters are extracted or observed on-the-fly from the input speech signal. They are then encoded to a HTTP byte stream and sent in a streaming fashion to the server before the silence is detected—i.e. sent to server side system **180** before the utterance is complete. This aspect of the invention also facilitates a real-time behavior, since data can be transmitted and processed even while the user is still speaking.

Receive Answer from Server **243** is comprised of the following modules as shown in FIG. **3**: MS Agent **244**, Text-to-Speech Engine **245** and receive communication modules **246**. All three modules interact to receive the answer from server side system **180**. As illustrated in FIG. **3**, the receive communication process consists of three separate processes implemented as a receive routine on client side system **150**: a Receive the Best Answer **258** receives the best answer over data link **160**B (the HTTP communication channel). The answer is de-compressed at **259** and then the answer is passed by code **260** to the MS Agent **244**, where it is received by code portion **254**. A routine **255** then articulates the answer using text-to-speech engine **257**. Of course, the text can also be displayed for additional feedback purposes on a monitor used with client side system **150**. The text to speech engine uses a natural language voice data file **256** associated with it that is appropriate for the particular language application (i.e., English, French, German, Japanese, etc.). As explained earlier when the answer is something more than text, it can be treated as desired to provide responsive information to the user, such as with a graphics image, a sound, a video clip, etc.

Uninitialization

The un-initialization routines and processes are illustrated in FIG. **4**. Three functional modules are used for un-initializing the primary components of the client side system **150**; these include SRE **270**, Communications **271** and MS Agent **272** un-initializing routines. To un-initialize SRE **220**A, memory that was allocated in the initialization phase is de-allocated by code **273** and objects created during such initialization phase are deleted by code **274**. Similarly, as illustrated in FIG. **4**, to un-initialize Communications module **220**C the INTERNET connection previously established with the server is closed by code portion **275** of the Communication Un-initialization routine **271**. Next the INTERNET session created at the time of initialization is also closed by routine **276**. For the un-initialization of the MS Agent **220**B, as illustrated in FIG. **4**, MS Agent Uninitialization routine **272** first releases the Commands Interface **227** using routine **277**. This releases the commands added to the property sheet during loading of the agent character by routine **225**. Next the Character Interface initialized by routine **226** is released by routine **278** and the Agent is unloaded at **279**. The Sink Object Interface is then also released **280** followed by the release of the Property Sheet Interface **281**. The Agent Notify Sink **282** then un-registers the Agent and finally the Agent Interface **283** is released which releases all the resources allocated during initialization steps identified in FIG. **2**D.

It will be appreciated by those skilled in the art that the particular implementation for such un-initialization processes and routines in FIG. **4** will vary from client platform to client platform, as for the other routines discussed above. The structure, operation, etc. of such routines are well-known in the art, and they can be implemented using a number of fairly straightforward approaches without undue effort. Accordingly, they are not discussed in detail herein.

Description of Server Side System **180**

Introduction

A high level flow diagram of the set of preferred processes implemented on server side system **180** of Natural Language Query System **100** is illustrated in FIGS. **11**A through FIG. **11**C. In a preferred embodiment, this process consists of a two step algorithm for completing the processing of the speech input signal, recognizing the meaning of the user's query, and retrieving an appropriate answer/response for such query.

The 1st step as illustrated in FIG. **11**A can be considered a high-speed first-cut pruning mechanism, and includes the following operations: after completing processing of the speech input signal, the user's query is recognized at step **1101**, so that the text of the query is simultaneously sent to Natural Language Engine **190** (FIG. **1**) at step **1107**, and to DB Engine **186** (also FIG.**1**) at step **1102**. By "recognized" in this context it is meant that the user's query is converted into a text string of distinct native language words through the HMM technique discussed earlier.

At NLE **190**, the text string undergoes morphological linguistic processing at step **1108**: the string is tokenized the

tags are tagged and the tagged tokens are grouped Next the noun phrases (NP) of the string are stored at **1109**, and also copied and transferred for use by DB Engine **186** during a DB Process at step **1110**. As illustrated in FIG. **11**A, the string corresponding to the user's query which was sent to the DB Engine **186** at **1102**, is used together with the NP received from NLE **190** to construct an SQL Query at step **1103**. Next, the SQL query is executed at step **1104**, and a record set of potential questions corresponding to the user's query are received as a result of a full-text search at **1105**, which are then sent back to NLE **190** in the form of an array at step **1106**.

As can be seen from the above, this first step on the server side processing acts as an efficient and fast pruning mechanism so that the universe of potential "hits" corresponding to the user's actual query is narrowed down very quickly to a manageable set of likely candidates in a very short period of time.

Referring to FIG. **11**B, in contrast to the first step above, the 2^{nd} step can be considered as the more precise selection portion of the recognition process. It begins with linguistic processing of each of the stored questions in the array returned by the full-text search process as possible candidates representing the user's query. Processing of these stored questions continues in NLE **190** as follows: each question in the array of questions corresponding to the record set returned by the SQL full-text search undergoes morphological linguistic processing at step **1111**: in this operation, a text string corresponding to the retrieved candidate question is tokenized, the tags are tagged and the tagged tokens are grouped. Next, noun phrases of the string are computed and stored at step **1112**. This process continues iteratively at point **1113**, and the sequence of steps at **1118**,**1111**, **1112**, **1113** are repeated so that an NP for each retrieved candidate question is computed and stored. Once an NP is computed for each of the retrieved candidate questions of the array, a comparison is made between each such retrieved candidate question and the user's query based on the magnitude of the NP value at step **1114**. This process is also iterative in that steps **1114**, **1115**, **1116**, **1119** are repeated so that the comparison of the NP for each retrieved candidate question with that of the NP of the user's query is completed. When there are no more stored questions in the array to be processed at step **1117**, the stored question that has the maximum NP relative to the user's query, is identified at **1117**A as the stored question which best matches the user's query.

Notably, it can be seen that the second step of the recognition process is much more computationally intensive than the first step above, because several text strings are tokenized, and a comparison is made of several NPs. This would not be practical, nonetheless, if it were not for the fact that the first step has already quickly and efficiently reduced the candidates to be evaluated to a significant degree. Thus, this more computationally intensive aspect of the present invention is extremely valuable, however because it yields extremely high accuracy in the overall query recognition process. In this regard, therefore, this second step of the query recognition helps to ensure the overall accuracy of the system, while the first step helps to maintain a satisfactory speed that provides a real-time feel for the user.

As illustrated in FIG. **11**C, the last part of the query/response process occurs by providing an appropriate matching answer/response to the user. Thus, an identity of a matching stored question is completed at step **1120**. Next a file path corresponding to an answer of the identified matching question is extracted at step **1121**. Processing continues so that the answer is extracted from the file path at **1122** and finally the answer is compressed and sent to client side system **150** at step **1123**.

The discussion above is intended to convey a general overview of the primary components, operations, functions and characteristics of those portions of NLQS system **100** that reside on server side system **180**. The discussion that follows describes in more detail the respective sub-systems.

Software Modules Used in Server Side System **180**

The key software modules used on server-side system **180** of the NLQS system are illustrated in FIG. **5**. These include generally the following components: a Communication module **500**—identified as CommunicationServer ISAPI **500**A (which is executed by SRE Server-side **182**—FIG. **1** and is explained in more detail below), and a database process DBProcess module **501** (executed by DB Engine **186**—FIG. **1**). Natural language engine module **500**C (executed by NLE **190**—FIG.**1**) and an interface **500**B between the NLE process module **500**C and the DBProcess module **500**B. As shown here, CommunicationServerISAPI **500**A includes a server-side speech recognition engine and appropriate communication interfaces required between client side system **150** and server side system **180**. As further illustrated in FIG. **5**, server-side logic of Natural Language Query System **100** also can be characterized as including two dynamic link library components: CommunicationServerISAPI **500** and DBProcess **501**. The CommunicationServerIASPI **500** is comprised of 3 sub-modules: Server-side Speech Recognition Engine module **500**A; Interface module **500**B between Natural Language Engine modules **500**C and DBProcess **501**; and the Natural Language Engine modules **500**C.

DB Process **501** is a module whose primary function is to connect to a SQL database and to execute an SQL query that is composed in response to the user's query. In addition, this module interfaces with logic that fetches the correct answer from a file path once this answer is passed to it from the Natural Language Engine module **500**C.

Speech Recognition Sub-System **182** on Server-Side System **180**

The server side speech recognition engine module **500**A is a set of distributed components that perform the necessary functions and operations of speech recognition engine **182** (FIG.**1**) at server-side **180**. These components can be implemented as software routines that are executed by server side **180** in conventional fashion. Referring to FIG. **4**A, a more detailed break out of the operation of the speech recognition components **600** at the server-side can be seen as follows:

Within a portion **601** of the server side SRE module **500**A, the binary MFCC vector byte stream corresponding to the speech signal's acoustic features extracted at client side system **150** and sent over the communication channel **160** is received. The MFCC acoustic vectors are decoded from the encoded HTTP byte stream as follows: Since the MFCC vectors contain embedded NULL characters, they cannot be transferred in this form to server side system **180** as such using HTTP protocol. Thus the MFCC vectors are first encoded at client-side **150** before transmission in such a way that all the speech data is converted into a stream of bytes without embedded NULL characters in the data. At the very end of the byte stream a single NULL character is introduced to indicate the termination of the stream of bytes to be transferred to the server over the INTERNET **160**A using HTTP protocol.

As explained earlier, to conserve latency time between the client and server, a smaller number of bytes (just the 13 MFCC coefficients) are sent from client side system **150** to

server side system **180**. This is done automatically for each platform to ensure uniformity, or can be tailored by the particular application environment—i.e., such as where it is determined that it will take less time to compute the delta and acceleration coefficients at the server (26 more calculations), than it would take to encode them at the client, transmit them, and then decode them from the HTTP stream. In general, since server side system **180** is usually better equipped to calculate the MFCC delta and acceleration parameters, this is a preferable choice. Furthermore, there is generally more control over server resources compared to the client's resources, which means that future upgrades, optimizations, etc., can be disseminated and shared by all to make overall system performance more reliable and predictable. So, the present invention can accommodate even the worst-case scenario where the client's machine may be quite thin and may just have enough resources to capture the speech input data and do minimal processing.

Dictionary Preparation & Grammar Files

Referring to FIG. **4**A, within code block **605**, various options selected by the user (or gleaned from the user's status within a particular application) are received. For instance, in the case of a preferred remote learning system, Course, Chapter and/or Section data items are communicated. In the case of other applications (such as e-commerce) other data options are communicated, such as the Product Class, Product Category, Product Brand, etc. loaded for viewing within his/her browser. These selected options are based on the context experienced by the user during an interactive process, and thus help to limit and define the scope—i.e. grammars and dictionaries that will be dynamically loaded to speech recognition engine **182** (FIG. **1**) for Viterbi decoding during processing of the user speech utterance. For speech recognition to be optimized both grammar and dictionary files are used in a preferred embodiment. A Grammar file supplies the universe of available user queries; i.e., all the possible words that are to be recognized. The Dictionary file provides phonemes (the information of how a word is pronounced—this depends on the specific native language files that are installed—for example, UK English or US English) of each word contained in the grammar file. It is apparent that if all the sentences for a given environment that can be recognized were contained in a single grammar file then recognition accuracy would be deteriorated and the loading time alone for such grammar and dictionary files would impair the speed of the speech recognition process.

To avoid these problems, specific grammars are dynamically loaded or actively configured as the current grammar according to the user's context, i.e., as in the case of a remote learning system, the Course, Chapter and/or Section selected. Thus the grammar and dictionary files are loaded dynamically according to the given Course, Chapter and/or Section as dictated by the user, or as determined automatically by an application program executed by the user.

The second code block **602** implements the initialization of Speech Recognition engine **182** (FIG.**1**). The MFCC vectors received from client side system **150** along with the grammar filename and the dictionary file names are introduced to this block to initialize the speech decoder.

As illustrated in FIG. **4**A, the initialization process **602** uses the following sub-routines: A routine **602***a* for loading an SRE library. This then allows the creation of an object identified as External Source with code **602***b* using the received MFCC vectors. Code **602***c* allocates memory to hold the recognition objects. Routine **602***d* then also creates and initializes objects that are required for the recognition such as: Source, Coder, Recognizer and Results Loading of the Dictionary created by code **602***e*, Hidden Markov Models (HMMs) generated with code **602***f*; and Loading of the Grammar file generated by routine **602***g*.

Speech Recognition **603** is the next routine invoked as illustrated in FIG. **4**A, and is generally responsible for completing the processing of the user speech signals input on the client side **150**, which, as mentioned above, are preferably only partially processed (i.e., only MFCC vectors are computed during the first phase) when they are transmitted across link **160**. Using the functions created in External Source by subroutine **602***b*, this code reads MFCC vectors, one at a time from an External Source **603***a*, and processes them in block **603***b* to realize the words in the speech pattern that are symbolized by the MFCC vectors captured at the client. During this second phase, an additional 13 delta coefficients and an additional 13 acceleration coefficients are computed as part of the recognition process to obtain a total of 39 observation vectors O_t referred to earlier. Then, using a set of previously defined Hidden Markov Models (HMMs), the words corresponding to the user's speech utterance are determined in the manner described earlier. This completes the word "recognition" aspect of the query processing, which results are used further below to complete the query processing operations.

It will be appreciated by those skilled in the art that the distributed nature and rapid performance of the word recognition process, by itself, is extremely useful and may be implemented in connection with other environments that do not implicate or require additional query processing operations. For example, some applications may simply use individual recognized words for filling in data items on a computer generated form, and the aforementioned systems and processes can provide a rapid, reliable mechanism for doing so. Once the user's speech is recognized, the flow of SRE **182** passes to Un-initialize SRE routine **604** where the speech engine is un-initialized as illustrated. In this block all the objects created in the initialization block are deleted by routine **604***a*, and memory allocated in the initialization block during the initialization phase are removed by routine **604***b*.

Again, it should be emphasized that the above are merely illustrative of embodiments for implementing the particular routines used on a server side speech recognition system of the present invention. Other variations of the same that achieve the desired functionality and objectives of the present invention will be apparent from the present teachings.

Database Processor **186** Operation—DBProcess

Construction of an SQL Query used as part of the user query processing is illustrated in FIG. **4**B, a SELECT SQL statement is preferably constructed using a conventional CONTAINS predicate. Module **950** constructs the SQL query based on this SELECT SQL statement, which query is used for retrieving the best suitable question stored in the database corresponding to the user's articulated query, (designated as Question here). A routine **951** then concatenates a table name with the constructed SELECT statement. Next, the number of words present in each Noun Phrase of Question asked by the user is calculated by routine **952**. Then memory is allocated by routine **953** as needed to accommodate all the words present in the NP. Next a word List (identifying all the distinct words present in the NP) is obtained by routine **954**. After this, this set of distinct words are concatenated by routine **955** to the SQL Query separated with a NEAR ( ) keyword. Next, the AND keyword is concatenated to the SQL Query by routine **956** after each NP. Finally memory resources are freed by code **957** so as to

allocate memory to store the words received from NP for any next iteration. Thus, at the end of this process, a completed SQL Query corresponding to the user's articulated question is generated.

Connection to SQL Server—As illustrated in FIG. **4**C, after the SQL Query is constructed by routine **710**, a routine **711** implements a connection to the query database **717** to continue processing of the user query. The connection sequence and the subsequent retrieved record set is implemented using routines **700** which include the following:

1. Server and database names are assigned by routine **711**A to a DBProcess member variable
2. A connection string is established by routine **711**B;
3. The SQL Server database is connected under control of code **711**C
4. The SQL Query is received by routine **712**A
5. The SQL Query is executed by code **712**B
6. Extract the total number of records retrieved by the query—**713**
7. Allocate the memory to store the total number of paired questions—**713**
8. Store the entire number of paired questions into an array—**713**

Once the Best Answer ID is received at **716** FIG. **4**C, from the NLE **14** (FIG. **5**), the code corresponding **716**C receives it passes it to code in **716**B where the path of the Answer file is determined using the record number. Then the file is opened **716**C using the path passed to it and the contents of the file corresponding to the answer is read. Then the answer is compressed by code in **716**D and prepared for transmission over the communication channel **160**B (FIG. **1**).

NLQS Database **188**—Table Organization

FIG. **6** illustrates a preferred embodiment of a logical structure of tables used in a typical NLQS database **188** (FIG.**1**). When NLQS database **188** is used as part of NLQS query system **100** implemented as a remote learning/training environment, this database will include an organizational multi-level hierarchy that consists typically of a Course **701**, which is made of several chapters **702**, **703**, **704**. Each of these chapters can have one or more Sections **705**, **706**, **707** as shown for Chapter **1**. A similar structure can exist for Chapter **2**, Chapter **3** . . . Chapter N. Each section has a set of one or more question—answer pairs **708** stored in tables described in more detail below. While this is an appropriate and preferable arrangement for a training/learning application, it is apparent that other implementations would be possible and perhaps more suitable for other applications such as e-commerce, e-support, INTERNET browsing, etc., depending on overall system parameters.

It can be seen that the NLQS database **188** organization is intricately linked to the switched grammar architecture described earlier. In other words, the context (or environment) experienced by the user can be determined at any moment in time based at the selection made at the section level, so that only a limited subset of question-answer pairs **708** for example are appropriate for section **705**. This in turn means that only a particular appropriate grammar for such question-answer pairs may be switched in for handling user queries while the user is experiencing such context. In a similar fashion, an e-commerce application for an INTERNET based business may consist of a hierarchy that includes a first level "home" page **701** identifying user selectable options (product types, services, contact information, etc.), a second level may include one or more "product types" pages **702**, **703**, **704**, a third page may include particular product models **705**, **706**, **707**, etc., and with appropriate question-answer pairs **708** and grammars customized for handling queries for such product models. Again, the particular implementation will vary from application to application, depending on the needs and desires of such business, and a typical amount of routine optimization will be necessary for each such application.

Table Organization

In a preferred embodiment, an independent database is used for each Course. Each database in turn can include three types of tables as follows: a Master Table as illustrated in FIG. **7**A, at least one Chapter Table as illustrated in FIG. **7**B and at least one Section Table as illustrated in FIG. **7**C.

As illustrated in FIG. **7**A, a preferred embodiment of a Master Table has six columns—Field Name **70**A, Data Type **702**A, Size **703**A, Null **704**A, Primary Key **705**A and Indexed **706**A. These parameters are well-known in the art of database design and structure. The Master Table has only two fields—Chapter Name **707**A and Section Name **708**A. Both ChapterName and Section Name are commonly indexed.

A preferred embodiment of a Chapter Table is illustrated in FIG. **7**B. As with the Master Table, the Chapter Table has six (6) columns—Field Name **720**, Data Type **721**, Size **722**, Null **723**, Primary Key **724** and Indexed **725**. There are nine (9) rows of data however, in this case,—Chapter_ID **726**, Answer_ID **727**, Section Name **728**, Answer_Title **729**, PairedQuestion **730**, AnswerPath **731**, Creator **732**, Date of Creation **733** and Date of Modification **734**.

An explanation of the Chapter Table fields is provided in FIG. **7**C. Each of the eight (8) Fields **720** has a description **735** and stores data corresponding to:

- AnswerID **727**—an integer that is automatically incremented for each answer given for user convenience
- Section_Name **728**—the name of the section to which the particular record belongs. This field along with the AnswerID is used as the primary key
- Answer_Title **729**—A short description of the title of the answer to the user query
- PairedQuestion **730**—Contains one or more combinations of questions for the related answers whose path is stored in the next column AnswerPath
- AnswerPath **731**—contains the path of a file, which contains the answer to the related questions stored in the previous column; in the case of a pure question/answer application, this file is a text file, but, as mentioned above, could be a multi-media file of any kind transportable over the data link **160**
- Creator **732**—Name of Content Creator
- Date_of_Creation **733**—Date on which content was created
- Date of Modification **734**—Date on which content was changed or modified

A preferred embodiment of a Section Table is illustrated in FIG. **7**D. The Section Table has six (6) columns—Field Name **740**, Data Type **741**, Size **742**, Null **743**, Primary Key **744** and Indexed **745**. There are seven (7) rows of data—Answer_ID **746**, Answer_Tide **747**, PairedQuestion **748**, AnswerPath **749**, Creator **750**, Date of Creation **751** and Date of Modification **752**. These names correspond to the same fields, columns already described above for the Master Table and Chapter Table.

Again, this is a preferred approach for the specific type of learning/training application described herein. Since the number of potential applications for the present invention is quite large, and each application can be customized, it is expected that other applications (including other learning/

training applications) will require and/or be better accommodated by another table, column, and field structure/hierarchy.

Search Service and Search Engine—A query text search service is performed by an SQL Search System **1000** shown in FIG. **10**. This system provides querying support to process fill-text searches. This is where full-text indexes reside.

In general, SQL Search System determines which entries in a database index meet selection criteria specified by a particular text query that is constructed in accordance with an articulated user speech utterance. The Index Engine **1011**B is the entity that populates the Full-Text Index tables with indexes which correspond to the indexable units of text for the stored questions and corresponding answers. It scans through character strings, determines word boundaries, removes all noise words and then populates the full-text index with the remaining words. For each entry in the full text database that meets the selection criteria, a unique key column value and a ranking value are returned as well. Catalog set **1013** is a file-system directory that is accessible only by an Administrator and Search Service **1010**. Full-text indexes **1014** are organized into full-text catalogs, which are referenced by easy to handle names. Typically, full-text index data for an entire database is placed into a single full-text catalog.

The schema for the full-text database as described (FIG. **7**, FIG. **7**A, FIG. **7**B, FIG. **7**C, FIG. **7**D) is stored in the tables **1006** shown in FIG. **10**. Take for example, the tables required to describe the structure the stored question/answer pairs required for a particular course. For each table Course Table, Chapter Table, Section Table, there are fields—column information that define each parameters that make up the logical structure of the table. This information is stored in User and System tables **1006**. The key values corresponding to those tables are stored as Full-Text catalogs **1013**. So when processing a full-text search, the search engine returns to the SQL Server the key values of the rows that match the search criteria. The relational engine then uses this information to respond to the query.

As illustrated in FIG. **10**, a Full-Text Query Process is implemented as follows:

1. A query **1001** that uses a SQL full-text construct generated by DB processor **186** is submitted to SQL Relational Engine **1002**.
2. Queries containing either a CONTAINS or FREETEXT predicate are rewritten by routine **1003** so that a responsive rowset returned later from Full-Text Provider **1007** will be automatically joined to the table that the predicate is acting upon. This rewrite is a mechanism used to ensure that these predicates are a seamless extension to a traditional SQL Server. After the compiled query is internally rewritten and checked for correctness in item **1003**, the query is passed to RUN TIME module **1004**. The function of module **1004** is to convert the rewritten SQL construct to a validated run-time process before it is sent to the Full-Text Provider, **1007**.
3. After this, Full-Text Provider **1007** is invoked, passing the following information for the query:
   a. A ft_search_condition parameter (this is a logical flag indicating a full text search condition)
   b. A name of a full-text catalog where a full-text index of a table resides
   c. A locale ID to be used for language (for example, word breaking)
   d. Identities of a database, table, and column to be used in the query
   e. If the query is comprised of more than one full-text construct; when this is the case Full-text provider **1007** is invoked separately for each construct.
4. SQL Relational Engine **1002** does not examine the contents of ft_search_condition. Instead, this information is passed along to Full-text provider **1007**, which verifies the validity of the query and then creates an appropriate internal representation of the full-text search condition.
5. The query request/command **1008** is then passed to Querying Support **101**A.
6. Querying Support **1012** returns a rowset **1009** from Full-Text Catalog **1013** that contains unique key column values for any rows that match the full-text search criteria. A rank value also is returned for each row.
7. The rowset of key column values **1009** is passed to SQL Relational Engine **1002**. If processing of the query implicates either a CONTAINSTABLE( ) or FREETEXTTABLE( ) function, RANK values are returned; otherwise, any rank value is filtered out.
8. The rowset values **1009** are plugged into the initial query with values obtained from relational database **1006**, and a result set **1015** is then returned for further processing to yield a response to the user.

At this stage of the query recognition process, the speech utterance by the user has already been rapidly converted into a carefully crafted text query, and this text query has been initially processed so that an initial matching set of results can be further evaluated for a final determination of the appropriate matching question/answer pair. The underlying principle that makes this possible is the presence of a full-text unique key column for each table that is registered for full-text searches. Thus when processing a full-text search, SQL Search Service **1010** returns to SQL server **1002** the key values of the rows that match the database. In maintaining these full-text databases **1013** and full text indexes **1014**, the present invention has the unique characteristic that the full-text indices **1014** are not updated instantly when the full-text registered columns are updated. This operation is eliminated, again, to reduce recognition latency, increase response speed, etc. Thus, as compared to other database architectures, this updating of the full-text index tables, which would otherwise take a significant time, is instead done asynchronously at a more convenient time. Interface between NLE **190** and DB Processor **188**

The result set **1015** of candidate questions corresponding to the user query utterance are presented to NLE **190** for further processing as shown in FIG. **4**D to determine a "best" matching question/answer pair. An NLE/DBProcessor interface module coordinates the handling of user queries, analysis of noun-phrases (NPs) of retrieved questions sets from the SQL query based on the user query, comparing the retrieved question NPs with the user query NP, etc. between NLE **190** and DB Processor **188**. So, this part of the server side code contains functions, which interface processes resident in both NLE block **190** and DB Processor block **188**. The functions are illustrated in FIG. **4**D; As seen here, code routine **880** implements functions to extract the Noun Phrase (NP) list from the user's question. This part of the code interacts with NLE **190** and gets the list of Noun Phrases in a sentence articulated by the user. Similarly, Routine **813** retrieves an NP list from the list of corresponding candidate/paired questions **1015** and stores these questions into an (ranked by NP value) array. Thus, at this point, NP data has been generated for the user query, as well as for the candidate questions **1015**. As an example of determining the noun phrases of a sentence such as: "What issues have guided the President in considering the impact off foreign trade policy on American businesses?" NLE **190** would return the following as noun phrases: President,

issues, impact of foreign trade policy, American businesses, impact, impact of foreign trade, foreign trade, foreign trade policy, trade, trade policy, policy, businesses. The methodology used by NLE **190** will thus be apparent to those skilled in the art from this set of noun phrases and noun sub-phrases generated in response to the example query.

Next, a function identified as Get Best Answer ID **815** is implemented. This part of the code gets a best answer ID corresponding to the user's query. To do this, routines **813**A, **813**B first find out the number of Noun phrases for each entry in the retrieved set **1015** that match with the Noun phrases in the user's query. Then routine **815**a selects a final result record from the candidate retrieved set **1015** that contains the maximum number of matching Noun phrases.

Conventionally, nouns are commonly thought of as "naming" words, and specifically as the names of "people, places, or things". Nouns such as John, London, and computer certainly fit this description, but the types of words classified by the present invention as nouns is much broader than this. Nouns can also denote abstract and intangible concepts such as birth, happiness, evolution, technology, management, imagination, revenge, politics, hope, cookey, sport, and literacy. Because of the enormous diversity of nouns compared to other parts of speech, the Applicant has found that it is much more relevant to consider the noun phrase as a key linguistic metric. So, the great variety of items classified as nouns by the present invention helps to discriminate and identify individual speech utterances much easier and faster than prior techniques disclosed in the art.

Following this same thought, the present invention also adopts and implements another linguistic entity—the word phrase—to facilitate speech query recognition. The basic structure of a word phrase—whether it be a noun phrase, verb phrase, adjective phrase—is three parts—[pre-Head string],[Head] and [post-Head string]. For example, in the minimal noun phrase—"the children," "children" is classified as the Head of the noun phrase. In summary, because of the diversity and frequency of noun phrases, the choice of noun phrase as the metric by which stored answer is linguistically chosen, has a solid justification in applying this technique to the English natural language as well as other natural languages. So, in sum, the total noun phrases in a speech utterance taken together operate extremely well as unique type of speech query fingerprint.

The ID corresponding to the best answer corresponding to the selected final result record question is then generated by routine **815** which then returns it to DB Process shown in FIG.**4**C. As seen there, a Best Answer ID I is received by routine **716**A, and used by a routine **716**B to retrieve an answer file path. Routine **716**C then opens and reads the answer file, and communicates the substance of the same to routine **716**D. The latter then compresses the answer file data, and sends it over data link **160** to client side system **150** for processing as noted earlier (i.e., to be rendered into audible feedback, visual text/graphics, etc.). Again, in the context of a learning/instructional application, the answer file may consist solely of a single text phrase, but in other applications the substance and format will be tailored to a specific question in an appropriate fashion. For instance, an "answer" may consist of a list of multiple entries corresponding to a list of responsive category items (i.e., a list of books to a particular author) etc. Other variations will be apparent depending on the particular environment.

Natural Language Engine **190**

Again referring to FIG. **4**D, the general structure of NL engine **190** is depicted. This engine inplements the word analysis or morphological analysis of words that make up the user's query, as well as phrase analysis of phrases extracted from the query.

As illustrated in FIG. **9**, the functions used in a morphological analysis include tokenizers **802**A, stemmers **804**A and morphological analyzers **806**A. The functions that comprise the phrase analysis include tokenizers, taggers and groupers, and their relationship is shown in FIG. **8**.

Tokenizer **802**A is a software module that functions to break up text of an input sentence **801**A into a list of tokens **803**A. In performing this function, tokenizer **802**A goes through input text **801**A and treats it as a series of tokens or useful meaningful units that are typically larger than individual characters, but smaller than phrases and sentences. These tokens **803**A can include words, separable parts of word and punctuation. Each token **803**A is given an offset and a length. The first phase of tokenization is segmentation, which extracts the individual tokens from the input text and keeps track of the offset where each token originated from in the input text. Next, categories are associated with each token, based on its shape. The process of tokenization is well-known in the art, so it can be performed by any convenient application suitable for the present invention.

Following tokenization, a stemmer process **804**A is executed, which can include two separate forms—inflectional and derivational, for analyzing the tokens to determine their respective stems **805**A. An inflectional stemmer recognizes affixes and returns the word which is the stem. A derivational stemmer on the other hand recognizes derivational affixes and returns the root word or words. While stemmer **804**A associates an input word with its stem, it does not have parts of speech information. Analyzer **806**B takes a word independent of context, and returns a set of possible parts of speech **806**A.

As illustrated in FIG. **8**, phrase analysis **800** is the next step that is performed after tokenization. A tokenizer **802** generates tokens from input text **801**. Tokens **803** are assigned to parts of a speech tag by a tagger routine **804**, and a grouper routine **806** recognizes groups of words as phrases of a certain syntactic type. These syntactic types include for example the noun phrases mentioned earlier, but could include other types if desired such as verb phrases and adjective phrases. Specifically, tagger **804** is a parts-of-speech disambiguator, which analyzes words in context. It has a built-in morphological analyzer (not shown) that allows it to identify all possible parts of speech for each token. The output of tagger **804** is a string with each token tagged with a parts-of-speech label **805**. The final step in the linguistic process **800** is the grouping of words to form phrases **807**. This function is performed by the grouper **806**, and is very dependent, of course, on the performance and output of tagger component **804**.

Accordingly, at the end of linguistic processing **800**, a list of noun phrases (NP) **807** is generated in accordance with the user's query utterance. This set of NPs generated by NLE **190** helps significantly to refine the search for the best answer, so that a single-best answer can be later provided for the user's question.

The particular components of NLE **190** are shown in FIG. **4**D, and include several components. Each of these components implement the several different functions required in NLE **190** as now explained.

Initialize Grouper Resources Object and the Library **900**—this routine initializes the structure variables required to create grouper resource object and library. Specifically, it initializes a particular natural language used by NLE **190** to create a Noun Phrase, for example the English natural language is initialized for a system that serves the English

language market. In turn, it also creates the objects (routines) required for Tokenizer, Tagger and Grouper (discussed above) with routines **900**A, **900**B, **900**C and **900**D respectively, and initializes these objects with appropriate values. It also allocates memory to store all the recognized Noun Phrases for the retrieved question pairs.

Tokenizing of the words from the given text (from the query or the paired questions) is performed with routine **909**B—here all the words are tokenized with the help of a local dictionary used by NLE **190** resources. The resultant tokenized words are passed to a Tagger routine **909**C. At routine **909**C, tagging of all the tokens is done and the output is passed to a Grouper routine **909**D.

The Grouping of all tagged token to form NP list is implemented by routine **909**D so that the Grouper groups all the tagged token words and outputs the Noun Phrases.

Un-initializing of the grouper resources object and freeing of the resources, is performed by routines **909**EA, **909**EB and **909**EC. These include Token Resources, Tagger Resources and Grouper Resources respectively. After initialization, the resources are freed. The memory that was used to store all Noun Phrases are also de-allocated.

Additional Embodiments

In a c-commerce embodiment of the present invention as illustrated in FIG. **13**, a web page **1300** contains typical visible links such as Books **1310**, Music **1320** so that on clicking the appropriate link the customer is taken to those pages. The web page may be implemented using HTML, a Java applet, or similar coding techniques which interact with the user's browser. For example, if customer wants to buy an album C by Artist Albert, he traverses several web pages as follows: he first clicks on Music (FIG. **13**, **1360**), which brings up page **1400** where he/she then clicks on Records (FIG. **14**, **1450**). Alternatively, he/she could select CDs **1460**, Videos **1470**, or other categories of books **1410**, music **1420** or help **1430**. As illustrated in FIG. **15**, this brings up another web page **1500** with links for Records **1550**, with sub-categories—Artist **1560**, Song **1570**, Title **1580**, Genre **1590**. The customer must then click on Artist **1560** to select the artist of choice. This displays another web page **1600** as illustrated in FIG. **16**. On this page the various artists **1650** are listed as illustrated—Albert **1650**, Brooks **1660**, Charlie **1670**, Whyte **1690** are listed under the category Artists **1650**. The customer must now click on Albert **1660** to view the albums available for Albert. When this is done, another web page is displayed as shown in FIG. **17**. Again this web page **1700** displays a similar look and feel, but with the albums available **1760**, **1770**, **1780** listed under the heading Tides **1750**. The customer can also read additional information **1790** for each album. This album information is similar to the liner notes of a shrink-wrapped album purchased at a retail store. One Album A is identified, the customer must click on the Album A **1760**. This typically brings up another text box with the information about its availability, price, shipping and handling charges etc.

When web page **1300** is provided with functionality of a NLQS of the type described above, the web page interacts with the client side and server side speech recognition modules described above. In this case, the user initiates an inquiry by simply clicking on a button designated Contact Me for Help **1480** (this can be a link button on the screen, or a key on the keyboard for example) and is then told by character **1440** about how to elicit the information required. If the user wants Album A by artist Albert, the user could articulate "Is Album A by Brooks available?" in much the same way they would ask the question of a human clerk at a brick and mortar facility. Because of the rapid recognition performance of the present invention, the user's query would be answered in real-time by character **1440** speaking out the answer in the user's native language. If desired, a readable word balloon **1490** could also be displayed to see the character's answer and so that save/print options can also be implemented. Similar appropriate question/answer pairs for each page of the website can be constructed in accordance with the present teachings, so that the customer is provided with an environment that emulates a normal conversational human-like question and answer dialog for all aspects of the web site. Character **1440** can be adjusted and tailored according to the particular commercial application, or by the user's own preferences, etc. to have a particular voice style (man, woman, young, old, etc.) to enhance the customer's experience.

In a similar fashion, an articulated user query might be received as part of a conventional search engine query, to locate information of interest on the INTERNET in a similar manner as done with conventional text queries. If a reasonably close question/answer pair is not available at the server side (for instance, if it does not reach a certain confidence level as an appropriate match to the user's question) the user could be presented with the option of increasing the scope so that the query would then be presented simultaneously to one or more different NLEs across a number of servers, to improve the likelihood of finding an appropriate matching question/answer pair. Furthermore, if desired, more than one "match" could be found, in the same fashion that conventional search engines can return a number of potential "hits" corresponding to the user's query. For some such queries, of course, it is likely that real-time performance will not be possible (because of the disseminated and distributed processing) but the advantage presented by extensive supplemental question/answer database systems may be desirable for some users.

It is apparent as well that the NLQS of the present invention is very natural and saves much time for the user and the e-commerce operator as well. In an e-support embodiment, the customer can retrieve information quickly and efficiently, and without need for a live customer agent. For example, at a consumer computer system vendor related support site, a sample diagnostic page might be presented for the user, along with a visible support character to assist him/her. The user could then select items from a "symptoms" page (i.e., a "monitor" problem, a "keyboard" problem, a "printer" problem, etc.) simply by articulating such symptoms in response to prompting from the support character. Thereafter, the system will direct the user on a real-time basis to more specific sub-menus, potential solutions, etc. for the particular recognized complaint. The use of a programmable character thus allows the web site to be scaled to accommodate a large number of hits or customers without any corresponding need to increase the number of human resources and its attendant training issues.

As an additional embodiment, the searching for information on a particular web site may be accelerated with the use of the NLQS of the present invention. Additionally, a significant benefit is that the information is provided in a user-friendly manner through the natural interface of speech. The majority of web sites presently employ lists of frequently asked questions which the user typically wades item by item in order to obtain an answer to a question or issue. For example, as displayed in FIG. **13**, the customer clicks on Help **1330** to initiate the interface with a set of lists. Other options include computer related items at **1370** and frequently asked questions (FAQ) at **1380**.

As illustrated in FIG. **18**, a web site plan for typical web page is displayed. This illustrates the number of pages that

have to be traversed in order to reach the list of Frequently-Asked Questions. Once at this page, the user has to scroll and manually identify the question that matches his/her query. This process is typically a laborious task and may or may not yield the information that answers the user's query. The present art for displaying this information is illustrated in FIG. **18**. This figure identifies how the information on a typical web site is organized: the Help link (FIG. **13**, **1330**) typically shown on the home page of the web page is illustrated shown on FIG. **18** as **1800**. Again referring to FIG. **18**, each sub-category of information is listed on a separate page. For example, **1810** lists sub-topics such as 'First Time Visitors', 'Search Tips', 'Ordering', 'Shipping', 'Your Account' etc. Other pages deal with 'Account information' **1860**, 'Rates and Policies' **1850** etc. Down another level, there are pages that deal exclusively with a sub-sub topics on a specific page such as 'First Time Visitors' **1960**, 'Frequently Asked Questions' **1950**, 'Safe Shopping Guarantee' **1940**, etc. So if a customer has a query that is best answered by going to the Frequently Asked Questions link, he or she has to traverse three levels of busy and cluttered screen pages to get to the Frequently Asked Questions page **1950**. Typically, there are many lists of questions **1980** that have to be manually scrolled through. While scrolling visually, the customer then has to visually and mentally match his or her question with each listed question. If a possible match is sighted, then that question is clicked and the answer then appears in text form which then is read.

In contrast, the process of obtaining an answer to a question using a web page enabled with the present NLQS can be achieved much less laboriously and efficiently. The user would articulate the word "Help" (FIG. **13**, **1330**). This would immediately cause a character (FIG. **13**, **1340**) to appear with the friendly response "May I be of assistance. Please state your question?". Once the customer states the question, the character would then perform an animation or reply "Thank you, I will be back with the answer soon". After a short period time (preferably not exceeding 5–7 seconds) the character would then speak out the answer to the user's question. As illustrated in FIG. **18** the answer would be the answer **1990** returned to the user in the form of speech is the answer that is paired with the question **1950**. For example, the answer **1990**: "We accept Visa, MasterCard and Discover credit cards", would be the response to the query **2000** "What forms of payments do you accept?"

Another embodiment of the invention is illustrated in FIG. **12**. This web page illustrates a typical website that employs NLQS in a web-based learning environment. As illustrated in FIG. **12**, the web page in browser **1200**, is divided into two or more frames. A character **1210** in the likeness of an instructor is available on the screen and appears when the student initiates the query mode either by speaking the word "Help" into a microphone (FIG. **2B**, **215**) or by clicking on the link 'Click to Speak' (FIG. **12**, **1280**). Character **1210** would then prompt the student to select a course **1220** from the drop down list **1230**. If the user selects the course 'CPlusPlus', the character would then confirm verbally that the course "CPlusPlus" was selected. The character would then direct the student to make the next selection from the drop-down list **1250** that contains the selections for the chapters **1240** from which questions are available. Again, after the student makes the selection, the character **1210** confirms the selection by speaking. Next character **1210** prompts the student to select 'Section' **1260** of the chapter from which questions are available from the drop down list **1270**. Again, after the student makes the selection, character **1210** confirms the selection by articulating the 'Section' **1260** chosen. As a prompt to the student, a list of possible questions appear in the list box **1291**. In addition, tips **1290** for using the system are displayed. Once the selections are all made, the student is prompted by the character to ask the question as follows: "Please ask your query now". The student then speaks his query and after a short period of time, the character responds with the answer preceded by the question as follows: "The answer to your question . . . is as follows: . . . ". This procedure allows the student to quickly retrieve answers to questions about any section of the course and replaces the tedium of consulting books, and references or indices. In short, it is can serve a number of uses from being a virtual teacher answering questions on-the-fly or a flash card substitute.

From preliminary data available to the inventors, it is estimate that the system can easily accommodate 100–250 question/answer pairs while still achieving a real-time feel and appearance to the user (i.e., less than 10 seconds of latency, not counting transmission) using the above described structures and methods. It is expected, of course, that these figures will improve as additional processing speed becomes available, and routine optimizations are employed to the various components noted for each particular environment.

Again, the above are merely illustrative of the many possible applications of the present invention, and it is expected that many more web-based enterprises, as well as other consumer applications (such as intelligent, interactive toys) can utilize the present teachings. Although the present invention has been described in terms of a preferred embodiment, it will be apparent to those skilled in the art that many alterations and modifications may be made to such embodiments without departing from the teachings of the present invention. It will also be apparent to those skilled in the art that many aspects of the present discussion have been simplified to give appropriate weight and focus to the more germane aspects of the present invention. The microcode and software routines executed to effectuate the inventive methods may be embodied in various forms, including in a permanent magnetic media, a non-volatile ROM, a CD-ROM, or any other suitable machine-readable format. Accordingly, it is intended that the all such alterations and modifications be included within the scope and spirit of the invention as defined by the following claims.

What is claimed is:

1. A machine executable program for assisting a computing system to effectuate distributed voice query recognition comprising:

a first audio signal receiving routine for receiving user speech utterance signals representing speech utterances to be recognized, said speech utterances including sentences comprised of one or more words; and

a first signal processing routine adapted to generate representative speech data values from said speech utterance signals, said representative speech data values being characterized by a first data content that is substantially inadequate by itself for permitting recognition of words articulated in said speech utterance; and

a formatting routine for rendering said representative speech data values into a transmission format suitable for transmission over a communications channel to a second processing routine executing on a separate computing system

wherein said representative speech data values are transmitted continuously during said speech utterances within streaming packets and without waiting for silence to be detected and/or said speech utterances to be completed; and

wherein said first data content in said representative speech data values is used by said second processing routine to compute additional data content that when combined with said first data content is sufficient for a speech recognition routine to complete recognition of words articulated in said speech utterance at said separate computing system

and further wherein an amount of said first data content transmitted is configured and can be varied for said speech utterances based on signal processing capabilities of the computing system and/or transmission characteristics of said communications channel.

2. The program of claim **1**, wherein said program works within a browser program executing on said computing system as part of a client-server based system.

3. The program of claim **1**, wherein said first signal processing routine is further adapted to handle a stream of continuous speech utterances, such that a plurality of representative speech data values are generated for each of said speech utterances in real-time.

4. The program of claim **1**, wherein said first data content is sufficiently small that said formatting routine can handle in real-time a continuous stream of representative speech data values that may be generated for a corresponding stream of speech utterances.

5. The program of claim **1**, wherein each of said representative speech data values corresponds to a separate cepstral coefficient value for a corresponding frequency component of said user speech utterance signals, and said first data content corresponds to a set of said frequency components spanning an audible speech frequency range.

6. The program of claim **5**, wherein said additional data content corresponds to a set of delta and acceleration coefficients computed from a corresponding set of said ceptstral coefficient values.

7. The program of claim **6**, wherein said additional data content is generated by said second processing routine with less latency than would that resulting if said additional data content were generated by said first signal processing routine.

8. The program of claim **1**, wherein signal processing functions required to generate said first data content and said additional data content can be allocated as needed between said first signal processing routine operating on a client device and said second signal processing routine operating on a server device based on a determination of computing resources available to said first and second signal processing routines respectively on a client device-by-device basis.

9. The program of claim **1**, wherein said first signal processing routine is a set of instructions executed by any one of a host microprocessor, an embedded processor, and/or a digital signal processor (DSP).

10. The program of claim **1**, wherein said first signal processing routine and said first audio signal are implemented as part of a desktop computing system, a portable computing system, a personal digital assistant (PDA), a cell-phone, and/or an electronic interactive toy.

11. A distributed voice recognition system comprising:

a sound processing circuit adapted to receive a speech utterance and to generate associated speech utterance signals therefrom; and

a first signal processing circuit adapted to generate a first set of speech data values from said speech utterance signals, said first set of speech data values being insufficient by themselves for permitting recognition of words articulated in said speech utterance; and

a transmission circuit for formatting and transmitting said first set of speech data values over a communications channel to a second signal processing circuit;

wherein said first set of speech data values are sent in a streaming fashion over said channel before silence is detected and/or said speech utterance is completed; and

said second signal processing circuit being configured to generate a second set of speech data values based on receiving and processing said speech data values during said speech utterance and before silence is detected, such that second set of speech data values contain sufficient information to be usable by a word recognition engine for recognizing words in said speech utterance;

and further wherein at least some words are recognized in real-time and output as text before said speech utterance is completed.

12. The system of claim **11**, wherein said second set of speech data values include said first set of speech data values and a derived set of speech data values, which derived set of speech data values are computed based on said first speech data values.

13. The system of claim **12**, wherein said first set of data values are MFCC vector coefficients, and said derived set of speech data values are MFCC delta coefficients and a MFCC acceleration coefficients derived from said MFCC vector coefficients.

14. The system of claim **11**, wherein said second set of speech data values can be generated by said second signal processing circuit in a time that is less than the combination of a first time which would be required by said first signal processing circuit to generate said second set of speech data values from said first set of speech data values combined with a second time which would be required by said transmission circuit to format and transmit said second set of speech data values.

15. The system of claim **11**, wherein said second set of speech data values can be generated by said second signal processing circuit in less time than that which would be required by said first signal processing circuit to generate said second set of speech data values from said first set of speech data values.

16. The system of claim **11**, wherein signal processing responsibilities of said first and second signal processing circuits are allocated such that said first signal processing circuit performs less than approximately ½ the required signal processing operations needed to convert said speech utterance signals into a form usable by a word recognition engine.

17. The system of claim **11**, wherein signal processing functions required to generate said first and second set of speech data values can be allocated between said first signal processing circuit and second signal processing circuit as needed based on computing resources available to said first and second signal processing circuits respectively.

18. The system of claim **11**, wherein signal processing functions required to generate said first and second set of speech data values can be allocated between said first signal processing circuit and second signal processing circuit as needed based on computing resources available to said first and second signal processing circuits respectively.

19. The system of claim **11**, wherein signal processing functions performed by said first signal processing circuit and second signal processing circuit are configured based on: (i) computing resources available to said first and second signal processing circuits; (ii) performance characteristics of said transmission circuit; and (iii) transmission latencies of said communications channel.

20. The system of claim **11**, wherein said first signal processing circuit is also configured to assist said second

signal processing circuit with signal processing computations required to generate said second set of speech data values.

21. The system of claim **11**, wherein said first set of speech data values represent the least amount of data that can used by said second signal processing circuit to generate said second set of data values usable for a word recognition process.

22. A system for recognizing speech information, comprising:

storage means for storing one or more words to be recognized by the system based on the speech information; and

means for capturing speech signals corresponding to the speech information; and

first processing means for generating partially recognized speech data from said speech signals, said first processing means performing a first signal processing operation on said speech signals, said first signal processing operation being insufficient to permit said partially recognized speech data to be correlated with said one or more words; and

second processing means for generating fully recognized speech data from said partially recognized speech data, using a second signal processing operation, such that said fully recognized speech data can be correlated with said one or more words, said second processing means being distinct and physically separated from said first processing means; and

third processing means for generating recognized sentence data from said one or more words using natural language processing operations including word phrase analysis performed on said one or more words;

a non-permanent data transmission connection coupling said first and second processing means;

transmitting means for transmitting said partially processed speech data signals from said first processing means through said non-permanent data transmission connection to said second processing means, said transmitting means using a continuously generated byte stream that is transmitted while a speech utterance is occurring and before silence is detected;

wherein the system recognizes a complete sentence included in the speech information based on said recognized sentence data and determines a best response to said complete sentence in real-time.

23. The system of claim **22**, wherein said first processing means is located at a client site, and said second processing means is located at a remote server site.

24. The system of claim **23**, wherein said storage means is also located at said server site.

25. The system of claim **22**, wherein said non-permanent connection is either a circuit switched or packet switched connection.

26. The system of claim **22**, wherein said non-permanent connection includes an intranet network linking said first and second signal processing means.

27. The system of claim **22**, wherein said non-permanent connection is a wireless communications channel.

28. The system of claim **22**, wherein said first signal processing operation includes an operation for extracting spectral parameter vectors from said speech signals.

29. The system of claim **28**, wherein said first signal processing operation further includes an operation for decomposing the spectral parameter vectors with a Mel-frequency transfer process to obtain MFCC coefficients for said spectral parameter vectors.

30. The system of claim **28**, wherein said partially recognized speech data comprises an observation vector O_t which includes said MFCC coefficients and delta and acceleration coefficients obtained from said spectral parameter vectors.

31. The system of claim **1**, wherein said partially recognized speech data includes an observation vector O_t, and second signal processing operation includes a Viterbi decoding operation for mapping a sequence of said observation vectors O_t with a speech data symbol.

32. The system of claim **10**, wherein said second signal processing operation further includes a text conversion operation for converting said speech data symbol into one or more text words.

33. The system of claim **22**, wherein said second processing means further utilizes environment variables for determining which of said one or more text words correspond to such speech information.

34. A method of performing speech recognition comprising the steps of:

(a) receiving user speech utterance signals representing speech utterances to be recognized, said speech utterances including sentences comprised of one or more words; and

(b) generating representative speech data values with a first computing device which by themselves are insufficient for completing recognition of said one or more words contained in said speech utterance signals; and

(c) formatting said representative speech data values into a transmission format suitable for transmission over a communications channel from said first computing device to a second computing device; and

wherein said representative speech data values are transmitted continuously during said speech utterances within streaming packets and without waiting for silence to be detected and/or said speech utterances to be completed; and

(d) performing a recognition of said one or more words at said second computing device using said representative speech data values and additional speech data values derived from said representative speech data values to generate recognized text;

(e) performing a natural language processing operation on said recognized text to determine a meaning associated with said sentences in real-time.

35. The method of claim **34**, wherein recognition of said one or more words is achieved with less latency than that resulting if said recognition were performed entirely by said first computing device.

36. The method of claim **34**, wherein signal processing functions required to perform said recognition can be allocated and configured between said first computing device and said second computing device as needed based on an evaluation of computing resources available to said first and second computing devices respectively.

37. The method of claim **34**, wherein said first computing device is part of a client computing system, and said second computing device is part of a server computing system, and said communications channel is a network.

38. A speech recognition program operating on a server coupled through a network to a client device, the program comprising:

a receiving routine for receiving speech data from the client device, said speech data being associated with a speech utterance from a user of the client device;

wherein said speech data has a data content that is adjusted based on signal processing capabilities of the

client device and resources available to the server for processing speech data;

further wherein said speech data is transmitted continuously by the client device during said speech utterance within streaming packets and without waiting for silence to be detected and/or said speech utterance to be completed;

a speech recognition processing routine, for recognizing words contained in said speech utterance;

wherein when said speech data from the client device is insufficient for recognizing words, additional speech related data is computed by the server to generate additional speech data to be combined with said speech data as an input to said speech recognition processing routine;

a natural language processing routine, for recognizing word sentences contained in said speech utterance by performing one or more natural language processing operations on words contained in said speech utterance.

39. The program of claim **38**, wherein said data content is also adjusted by considering transmission characteristics of the network, and a transmission speed capability of the client device.

40. The program of claim **38**, wherein a calibration routine automatically determines an optimal setting for said data content.

41. The program of claim **40**, wherein said calibration routine executes on the client device.

42. The program of claim **38**, wherein said speech data includes mel frequency cepstral coefficients (MFCCs), and said additional speech related data includes MFCC delta coefficients and a MFCC acceleration coefficients derived from said MFCC vector coefficients.

43. The program of claim **38**, wherein the network includes one of an INTERNET connection, a LAN connection, a wireless connection, or a combination thereof.

44. The program of claim **38**, wherein said words contained in said speech utterance are recognized in real-time by said speech recognition routine.

45. The program of claim **44**, wherein said word sentences contained in said speech utterance are recognized in real-time by said natural language processing routine.

46. The program of claim **38**, wherein the client device includes one of a desktop computer, a portable computer, a personal digital assistant, and a cell phone.

47. The program of claim **38**, wherein said natural language processing operations include word phrase analysis, including a noun phrase analysis.

48. A method of performing speech recognition at a server coupled to a client device through a network, the method comprising the steps of:

(a) determining signal processing capabilities of the client device available for processing speech data associated with a user speech utterance;

(b) determining resources available to the server for processing speech related data that is to be transmitted by the client device to the server;

(c) configuring a data content to be used for transmitting said speech related data through the network based on the results of steps (a) and (b);

(d) receiving said speech related data with said data content at the server continuously from the client device during said speech utterance within streaming packets and without waiting for silence to be detected and/or said speech utterance to be completed;

(e) processing said speech related data at the server so that additional speech related data is computed at the server from said speech data and is used to augment said speech related data when said speech related data contains acoustic feature data from said speech utterance that is insufficient for the server to perform accurate recognition of words contained in said speech utterance;

(f) generating a speech data observation vector suitable for processing by a speech recognition routine, said speech data observation vector being based on said speech related data as received from the client device, and/or said speech related data as augmented by the server;

(g) recognizing words contained in said speech utterance with a speech recognition engine using said speech data observation vector;

(h) recognizing word sentences contained in said speech utterance by performing one or more natural language processing operations on words contained in said speech utterance.

49. The method of claim **48**, wherein said data content is also configured by considering transmission characteristics of the network, and a transmission speed capability of the client device.

50. The method of claim **48**, further including a step of executing a calibration routine to automatically determine an optimal setting for said data content.

51. The method of claim **50**, wherein said calibration routine executes on the client device.

52. The method of claim **48**, wherein said acoustic feature data includes mel frequency cepstral coefficients (MFCCs), and said additional speech related data includes MFCC delta coefficients and a MFCC acceleration coefficients derived from said MFCC vector coefficients.

53. The method of claim **48**, wherein the network includes one of an INTERNET connection, a LAN connection, a wireless connection, or a combination thereof.

54. The method of claim **48**, wherein said words contained in said speech utterance are recognized in real-time by said speech recognition routine.

55. The method of claim **54**, wherein said word sentences contained in said speech utterance are recognized in real-time by said natural language processing routine, and a response is sent to the client device in real-time.

56. The method of claim **48**, wherein the client device includes one of a desktop computer, a portable computer, a personal digital assistant, and a cell phone.

57. The method of claim **48**, wherein said natural language processing operations include word phrase analysis, including a noun phrase analysis.

* * * * *

EXHIBIT 2

US006665640B1

(12) **United States Patent**
Bennett et al.

(10) **Patent No.: US 6,665,640 B1**
(45) **Date of Patent: Dec. 16, 2003**

(54) **INTERACTIVE SPEECH BASED LEARNING/ TRAINING SYSTEM FORMULATING SEARCH QUERIES BASED ON NATURAL LANGUAGE PARSING OF RECOGNIZED USER QUERIES**

(75) Inventors: **Ian M. Bennett**, Palo Alto, CA (US); **Bandi Ramesh Babu**, Anantapur Gulbarga (IN); **Kishor Morkhandikar**, Bangalore (IN); **Pallaki Gururaj**, Bangalore (IN)

(73) Assignee: **Phoenix Solutions, Inc.**, Palo Alto, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/439,173**

(22) Filed: **Nov. 12, 1999**

(51) **Int. Cl.**[7] **G10L 15/18**; G10L 15/22; G06F 17/30

(52) **U.S. Cl.** **704/257**; 704/270.1; 704/275; 707/4

(58) **Field of Search** 704/257, 270.1, 704/275; 707/4

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,473,904	A	9/1984	Suehiro et al.	381/36
4,587,670	A	5/1986	Levinson et al.	381/43
4,783,803	A	11/1988	Baker et al.	381/42
4,785,408	A	11/1988	Britton et al.	

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

EP	1 094 388 A2	4/2001
EP	1 096 471 A1	5/2001
WO	9811534	3/1998

(List continued on next page.)

OTHER PUBLICATIONS

L. Travis, "Handbook of Speech Pathology," Appleton–Century–Crofts, Inc., 1957, pp. 91–124.

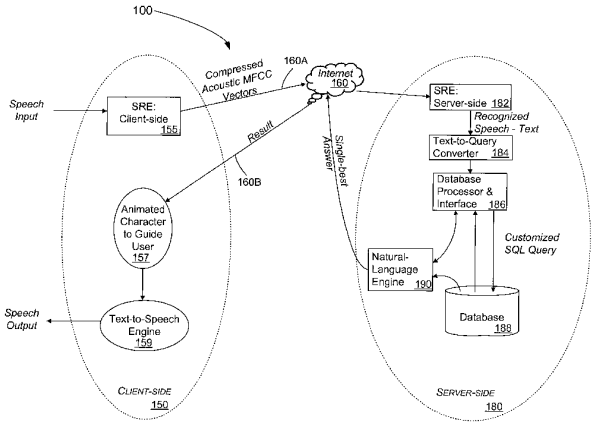
(List continued on next page.)

Primary Examiner—Tālivaldis Ivars Ŝmits
(74) *Attorney, Agent, or Firm*—J. Nicholas Gross

(57) **ABSTRACT**

A real-time speech-based learning/training system distributed between client and server, and incorporating speech recognition and linguistic processing for recognizing a spoken question and to provide an answer to the student in a learning or training environment implemented on an intranet or over the Internet, is disclosed. The system accepts the student's question in the form of speech at his or her computer, PDA or workstation where minimal processing extracts a sufficient number of acoustic speech vectors representing the utterance. The system as implemented accepts environmental variables such as course, chapter, section as selected by the user so that the search time, accuracy and response time for the question can be optimized. A minimum set of acoustic vectors extracted at the client are then sent via a communications channel to the server where additional acoustic vectors are derived. Using Hidden Markov Models (HMMs), and appropriate grammars and dictionaries conditioned by the course, chapter and section selections made by the student, the speech representing the user's query is fully decoding to text at the server. This text corresponding to the user's query is then simultaneously sent to a natural language engine and a database processor where an optimized SQL statement is constructed for a full-text search from a SQL database for a recordset of several stored questions that best matches the user's query. Further processing in the natural language engine narrows the search down to a single stored question. The answer that is paired to this single stored question is then retrieved from the file path and sent to the student computer in compressed form. At the student's computer, the answer is articulated using a text-to-speech engine in his or her native natural language. The system requires no training and can operate in several natural languages.

38 Claims, 31 Drawing Sheets



U.S. PATENT DOCUMENTS

4,852,170	A		7/1989	Bordeaux	
4,914,590	A		4/1990	Loatman et al.	364/419
4,991,094	A		2/1991	Fagan et al.	364/419
4,991,217	A		2/1991	Garrett et al.	381/43
5,068,789	A		11/1991	van Vliembergen	364/419
5,146,405	A		9/1992	Church	364/419
5,157,727	A		10/1992	Schloss	381/31
5,231,670	A		7/1993	Goldhor et al.	381/43
5,293,584	A		3/1994	Brown et al.	395/2.86
5,384,892	A		1/1995	Strong	395/2.52
5,475,792	A		12/1995	Stanford et al.	395/2.42
5,513,298	A		4/1996	Stanford et al.	395/2.52
5,540,589	A	*	7/1996	Waters	434/156
5,602,963	A		2/1997	Bissonnette et al.	395/2.84
5,615,296	A	*	3/1997	Stanford et al.	704/270.1
5,680,628	A		10/1997	Carus et al.	395/759
5,727,950	A		3/1998	Cook et al.	434/350
5,758,322	A		5/1998	Rongley	704/275
5,802,256	A		9/1998	Fawcett et al.	707/104
5,819,220	A		10/1998	Sarukkai et al.	704/243
5,867,817	A		2/1999	Catallo et al.	704/255
5,873,062	A		2/1999	Hansen et al.	
5,915,236	A		6/1999	Gould et al.	704/251
5,956,683	A		9/1999	Jacobs et al.	704/275
5,960,394	A		9/1999	Gould et al.	704/240
5,960,399	A		9/1999	Barclay et al.	704/270
5,983,190	A	*	11/1999	Trower, II et al.	704/276
5,995,928	A		11/1999	Nguyen et al.	
6,009,387	A		12/1999	Ramaswamy et al.	
6,029,124	A		2/2000	Gillick et al.	704/200
6,035,275	A		3/2000	Brode et al.	
6,119,087	A		9/2000	Kuhn et al.	
6,138,089	A		10/2000	Guberman	704/207
6,141,640	A		10/2000	Moo	
6,144,848	A		11/2000	Walsh et al.	455/419
6,144,938	A		11/2000	Surace et al.	704/257
6,256,607	B1		7/2001	Digalakis et al.	
6,269,336	B1		7/2001	Ladd et al.	
6,327,568	B1		12/2001	Joost	
6,356,869	B1	*	3/2002	Chapados et al.	704/275
6,363,349	B1		3/2002	Urs et al.	
6,374,219	B1		4/2002	Jiang	
6,381,594	B1		4/2002	Eichstaedt et al.	
6,389,389	B1		5/2002	Meunier et al.	
6,408,272	B1		6/2002	White et al.	
6,411,926	B1		6/2002	Chang	
6,427,063	B1		7/2002	Cook et al.	
2001/0032083	A1		10/2001	Van Cleven	
2001/0056346	A1		12/2001	Ueyama et al.	
2002/0046023	A1		4/2002	Fujji et al.	
2002/0059068	A1		5/2002	Rose et al.	
2002/0059069	A1		5/2002	Hsu et al.	
2002/0086269	A1		7/2002	Shpiro	
2002/0087325	A1		7/2002	Lee et al.	
2002/0087655	A1		7/2002	Bridgman et al.	
2002/0091527	A1		7/2002	Shiau	

FOREIGN PATENT DOCUMENTS

WO	WO 99/48011 A1	9/1999
WO	WO 00/14727 A1	3/2000
WO	WO 00/17854 A1	3/2000
WO	WO 00/20962 A2	4/2000
WO	WO 00/21075 A1	4/2000
WO	WO 00/21232 A2	4/2000
WO	WO 00/22610 A1	4/2000
WO	WO 00/30072 A2	5/2000
WO	WO 00/30287 A1	5/2000
WO	WO 00/68823 A2	11/2000
WO	WO 01/16936 A1	3/2001
WO	WO 01/18693 A2	3/2001
WO	WO 01/26093 A1	4/2001
WO	WO 01/78065 A1	10/2001
WO	WO 01/95312 A1	12/2001
WO	WO 02/03380 A1	1/2002

OTHER PUBLICATIONS

L.E. Baum, T. Petrie, "Statistical Inference for Probabilistic Functions for Finite State Markov Chains," *The Annals of Mathematical Statistics,* 37: 1554–1563, 1966.

P. Lieberman, "Intonation, Perception and Language," Research Monograph No. 38, MIT Press, Cambridge, Mass., 1967, pp. 5–37.

L.E. Baum et al, "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains," *The Annals of Mathematical Statistics,* 1970, vol. 41, No. 1, pp. 164–171.

J.L. Flanagan, "Speech Analysis Synthesis and Perception," 2nd edition, Springer–Verlag Berlin, 1972, pp. 1–53.

L.E. Baum, "An Inequality and Associated Maximation Technique in Statistical Estimation for Probabilistic Functions of Markov Processes," *Inequalities–III,* pp. 1–8, 1972.

G.D. Forney, "The Viterbi Algorithm," *Proceedings of the IEEE,* vol. 73, pp. 268–278, Mar. 1973.

J.H. Baker, "The Dragon System—An Overview," *IEEE Trans. on ASSP Processing,* ASSP–23(1): 24–29, Feb. 1975.

I. Bennett, "A Study of Speech Compression Using Analog Time Domain Sampling techniques," *A Dissertation Submitted to the Dept. Of Electrical Engineering and the Committee on Graduate Studies of Stanford University,* May 1975, pp. 16–32; 76–111.

H.R. Rabiner, "Digital Processing of Speech Signals," Prentice Hall, 198, pp. 116–171; 355–395.

F. Jelinek et al, "Continuous Speech Recognition Statistical Methods," *Handbook of Statistics,* vol. 2, P.R. Krishnaiah, Ed. Amsterdam. The Netherlands, North–Holland, 1982, pp. 549–573.

L.R. Bahl, F. Jeninek, R.L. Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI–5, pp. 179–190, Mar. 1983.

R.A. Hudson, "Word Grammar," Blackman Inc., Cambridge, MA, 1984, pp. 1–14; 41–42; 76–90; 94–98; 106–109; 211–220.

R. Quirk, S. Greenbaum, G. Leech and J. Svartvik, "A Comprehensive Grammar of English Language," Longman, London and New York, 1985, pp. 245–331.

J. Makhoul, S. Roucos, H. Gish, "Vector Quantization in Speech Coding," *Proceedings of the IEEE,* vol. 73, No. 11, Nov. 1985, pp. 1551–1588.

L. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE,* vol. 77, No. 2, Feb. 1989, pp. 257–286.

A. Gersho and R.M. Gray, "Vector Quantization and Compression," Kluwer Academic Publishers, 1991, pp. 309–340.

H.R. Rabiner and B.H. Juang, "Fundamentals of Speech Recognition," Prentice Hall, 1993, pp. 11–68.

Nelson Morgan, Herv´´ Bourlard, Steve Renals, Michael Cohen and Horacio Franco, "Hybrid Neural Network/Hidden Markov Model Systems for Continuous Speech Recognition," *Journal of Pattern Recognition and Artificial Intelligence,* vol. 7, No. 4, 1993, pp. 899–916.

L. Travis, "Handbook of Speech Pathology", Appleton–Century–Crofts, Inc., 1957.

L.E. Baum, T. Petrie, "Statistical inference for probablistic functions for finite state Markov chains", Ann. Math. Stat., 37: 1554–1563, 1966.

J.L. Flanagan, "Speech Analysis Synthesis and Perception", 2nd edition, Springer–Verlag Berlin, 1972.
L.E. Baum, "An inequality and associated maximation technique in statistical estimation for probablistic functions of Markov processes", Inequalities 3: 1–8, 1972.
R. Quirk, S. Greenbaum, G. Leech and J. Svartik, "A Comprehensive Grammar of English Language," Longman, London and New York, 1985, pp. 245–331.
J. Makhoul, S. Roucos, H. Gish, "Vector Quanttization in Speech Coding," *Proceedings of the IEEE,* vol. 73, No. 11, Nov. 1985, pp. 1551–1588.
L. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE,* vol. 77, No. 2, Feb. 1989, pp. 257–286.
A. Gersho and R.M. Gray, "Vector Quanttization and Compression," Kluwer Academic Publishers, 1991, pp. 309–340.
H.R. Rabiner and B.H. Juang, "Fundamentals of Speech Recognition," Prentice Hall, 1993, pp. 11–68.
Nelson Morgan, Herv´´ Bourlard, Steve Renals, Michael Cohen and Horacio Franco, "Hybrid Neural Network/Hidden Markov Model Systems for Continuous Speech Recognition," *Journal of Pattern Recognition and Artificial Intelligence,* vol. 7, No. 4, 1993, pp. 899–916.
G.D. Forney, "The Viterbi Algorithm," *Proceedings of the IEEE,* vol. 73, pp. 268–278, Mar. 1973.
J.H. Baker, "The Dragon System—An Overview," *IEEE Trans. on ASSP Processing,* ASSP–23(1): Feb. 24–29, 1975.
I. Bennett, "A Study of Speech Compression Using Analog Time Domain Sampling techniques," *A Dissertation Submitted to the Dept. Of Electrical Engineering and the Committee on Graduate Studies of Stanford University,* May 1975, pp. 16–32; 76–111.
H.R. Rabiner, "Digital Processing of Speech Signals," Prentice Hall, 1978, pp. 116–171; 355–395.
F. Jelinek et al, "Continuous Speech Recognition: Statistical Methods," *Handbook of Statistics,* vol. 2, P.R. Krishnaiah, Ed. Amsterdam, The Netherlands, North–Holland, 1982, pp. 549–573.
L.R. Bahl, F. Jeninek, R.L. Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI–5, pp. 179–190, Mar. 1983.
R.A. Hudson, "Word Grammar," Blackman Inc., Cambridge, MA, 1984, pp. 1–14; 41–42; 76–90; 94–98; 106–109; 211–220.
J.D. Ferguson, "Hidden Markov Analysis: An Introduction", in Hidden Markov Models for Speech, Institute of Defense Analyses, Princeton, NJ. 1980.
I. Bennett and J. Linvill, "A Study of Time Domain Speech Compression by Means of a new Analog Speech Processor", Journal of the Audio Engineering Society, vol. 23, No. 9, 1975.
R. Quirk, S. Greenbaum, G. Leech and J. Svartvik, A Comprehensive Grammar of English Language, Longman, London and New York, 1985.
I. Guyon and P. Wang editors *Advances in Pattern Recognition Systems using Neural Networks,* vol. 7 of a Series in Machine Perception and Artificial Intelligence, World Scientific, Feb. 1994.
P. Lieberman, "Intonation, Perception and Language", Research Monograph No. 38, MIT Press, Cambridge, Mass.
Arons, B., "The Design of Audio Servers and Toolkits for Supporting Speech in the User Interface," believed to be Published in: Journal of the American Voice I/O Society, pp. 27–41, Mar. 1991.
Hazen,T et al., "Recent Improvements in an Approach to Segment–Based Automatic Language Identification," believed to be published in: Proceedings of the 1994 International Conference on Spoken Language Processing, Yokohama, Japan, pp. 1883–1886, Sep. 1994.
House, D., "Spoken–Language Access to Multimedial (SLAM): A Multimodal Interface to the World–Wide–Web," Masters Thesis, Oregon Graduate Institute, Department of Computer Science & Engineering, 59 pages, Apr. 1995.
Julia, L. et al., "Http://www.speech SRI.com/Demos/ Atis.html," believed to be published in: Proceedings AAAI'97: Stanford, pp. 72–76, Jul. 1997.
Lau, R. et al, "Webgalaxy–Integrating Spoken Language and Hypertext Navigation," believed to be published in: in Kokkinakis, G. et al., (Eds.) Eurospeech '97, Proceedings of the 5th European Conference on Speech Communication and Technology, Rhodes (Greece), Sep. 22–25, 1997, pp. 883–886, 1997.
Digalakis, V. et al., "Product–Code Vector Quantization of Cepstral Parameters for Speech Recognition over the WWW," believed to be published in: Proc. ICSLP '98, 4 pages, 1998.
Melin, H., "On Word Boundary Detection in Digit–Based Speaker Verification," believed to be published in: Workshop on Speaker Recognition and its Commercial and Forensic Applications (RLA2C), Avignon, France, Apr. 20–23, pp. 46–49, 1998.
Ramaswamy, G. et al., "Compression of Acoustic Features for Speech Recognition in Network Environments," believed to be published in: IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 977–980, Jun. 1998.
Lu, B. et al., "Scalability Issues in the Real Time Protocol (RTP)," Project Report for CPSC 663 (Real Time Systems), Dept. of Computer Science, Texas A & M University, 19 pages, 1999.
Giuliani, D. et al., "Training of HMM with Filtered Speech Material for Hands–Free Recognition," believed to be published in: Proceedings of ICASSP '99, Phoenix, USA, 4 pages, 1999.
Digalakis,V. et al., "Quantization of Cepstral Parameters for Speech Recognition over the World Wide Web," believed to be published in: IEEE Journal on Selected Areas of Communications, 22 pages, 1999.
Tsakalidis, S. et al., "Efficient Speech Recognition Using Subvector Quantization and Discrete–Mixture HMMs," believed to be published in: Proc. ICASSP '99, 4 pages, 1999.
Lin, B. et al., "A Distributed Architecture for Cooperative Spoken Dialogue Agents with Coherent Dialogue State and History," believed to be published in: IEEE Automatic Speech Recognition and Understanding Workshop, Keystone, Colorado, USA, 4 pages, Dec. 1999.
Meunier, J., "RTP Payload Format for Distributed Speech Recognition," 48th IETF AVT WG—Aug. 3, 2000, 10 pages, 2000.
Sand Cherry Networks, SoftServer product literature, 2 pages, 2001.
Kim,H. et al., "A Bitstream–Based Front–End for Wireless Speech Recognition on IS–136 Communications System," IEEE Transactions on Speech and Audio Processing, vol. 9, No. 5, pp. 558–568, Jul. 2001.

* cited by examiner

Fig. 1

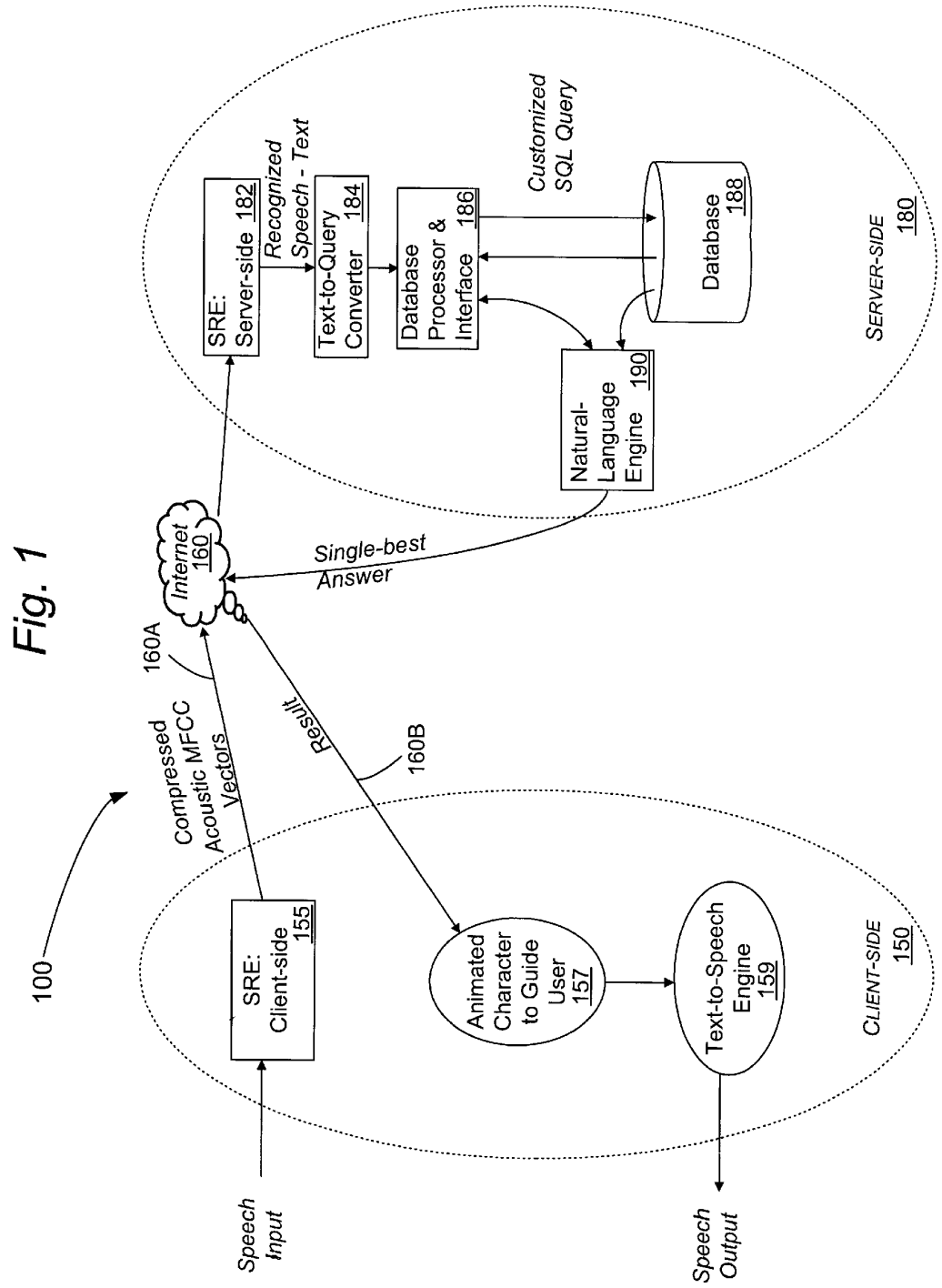

100
SRE: Client-side 155
Compressed Acoustic MFCC Vectors
160A
Internet 160
Result
160B
Single-best Answer
SRE: Server-side 182
Recognized Speech - Text
Text-to-Query Converter 184
Database Processor & Interface 186
Customized SQL Query
Database 188
Natural-Language Engine 190
SERVER-SIDE 180
Animated Character to Guide User 157
Text-to-Speech Engine 159
CLIENT-SIDE 150
Speech Input
Speech Output

Figure 2 (Page 1/3)
CLIENT-SIDE SYSTEM LOGIC

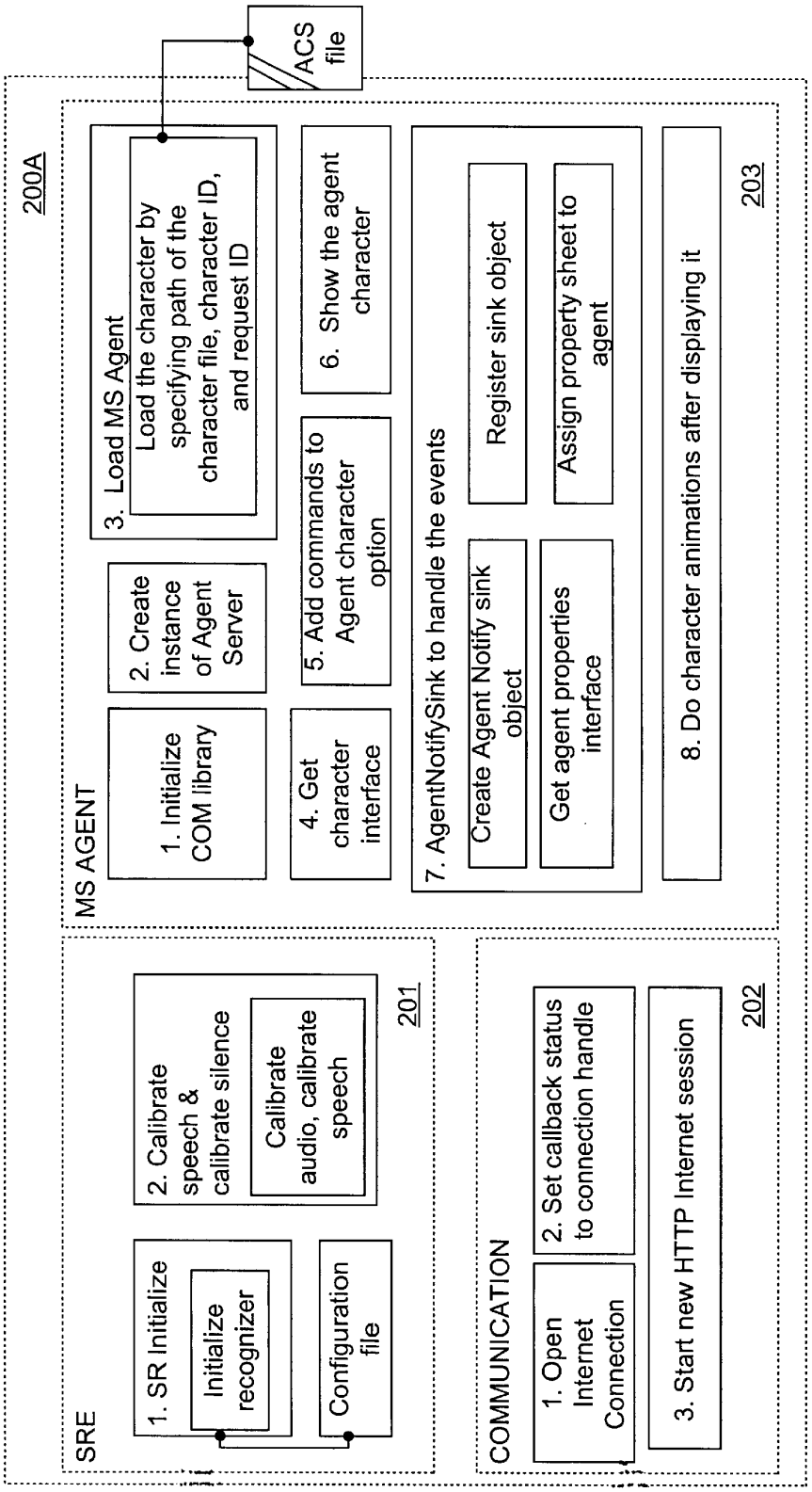
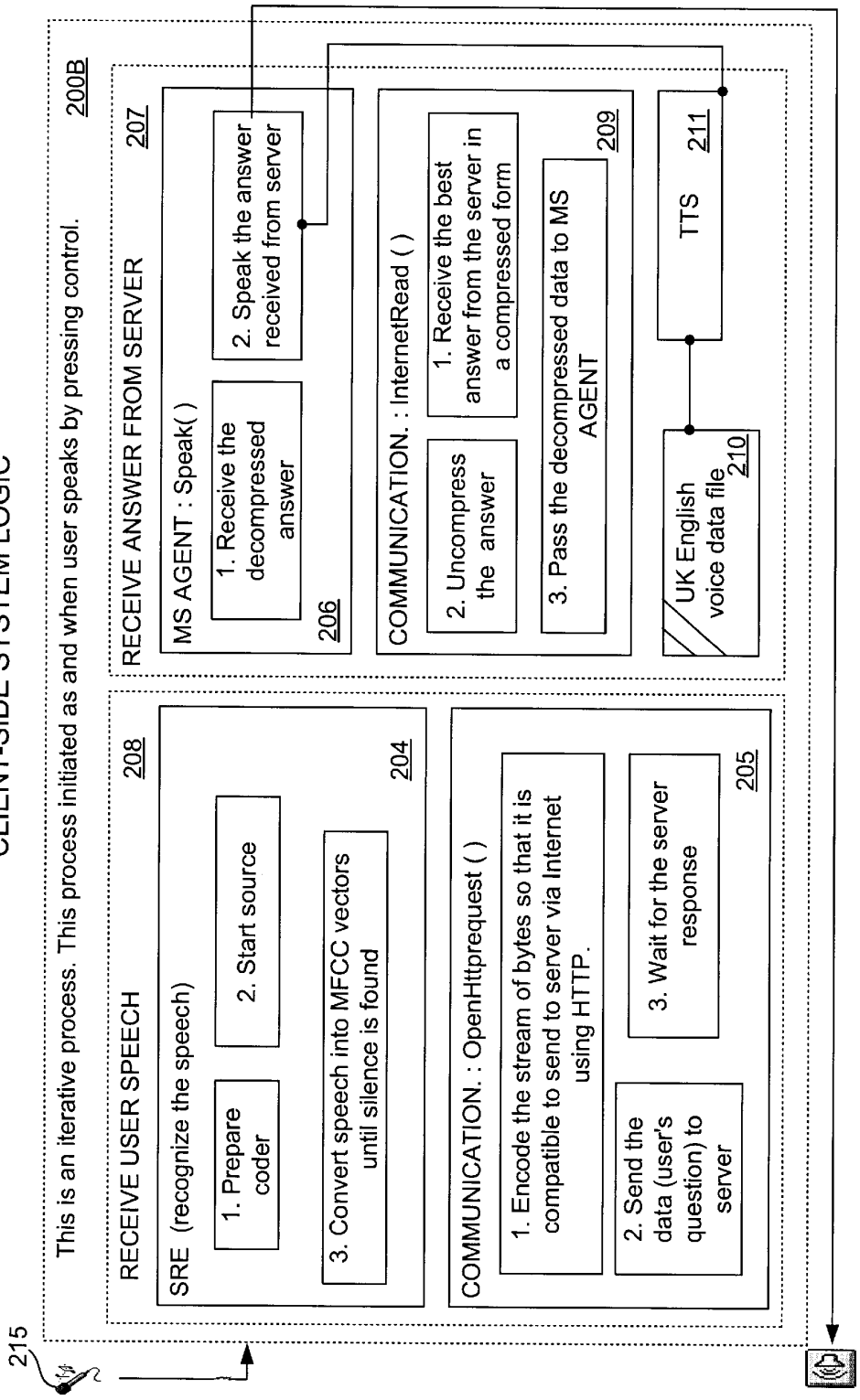


Figure 2 (Page 2/3)
CLIENT-SIDE SYSTEM LOGIC



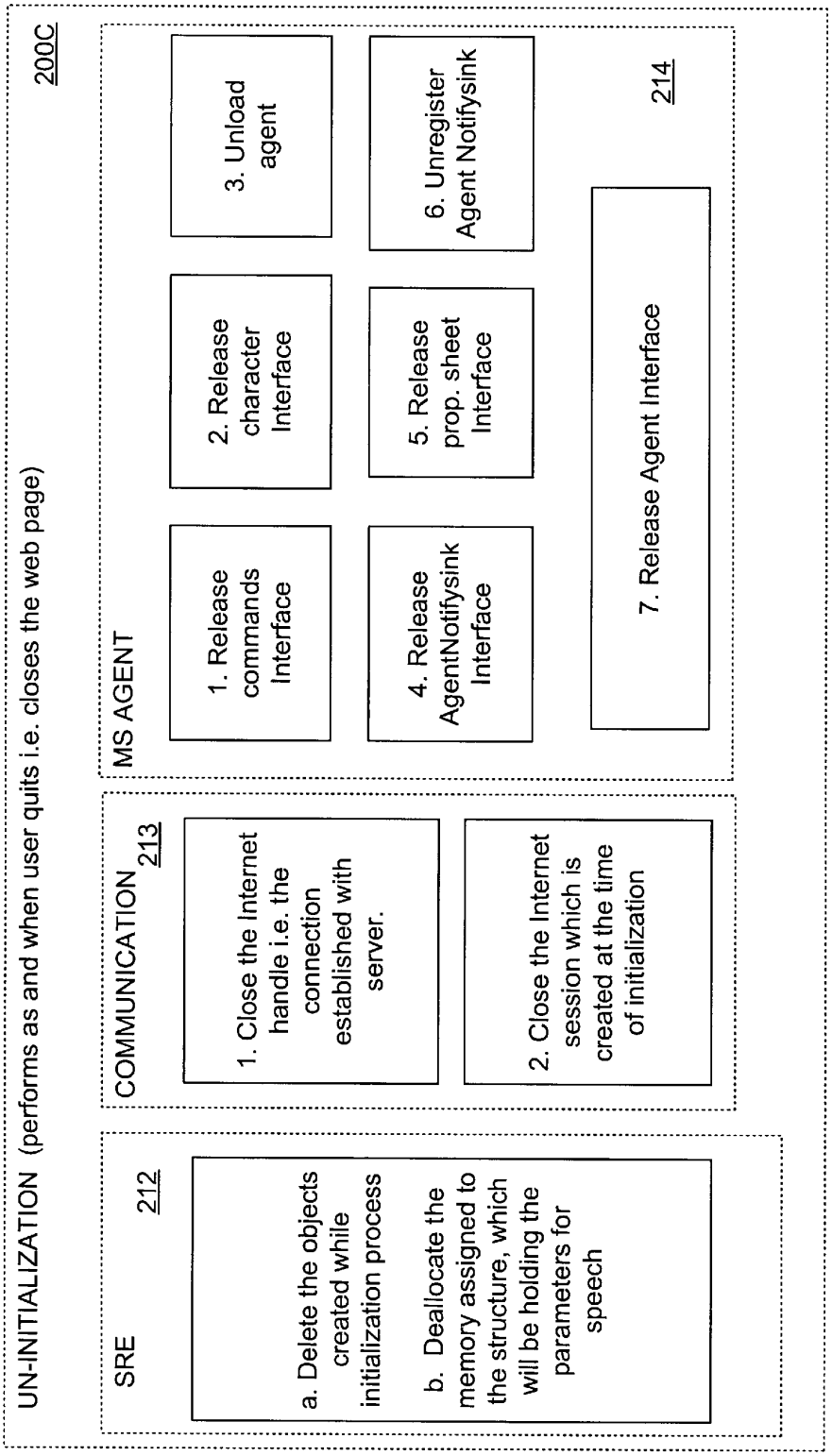


Figure 2 (Page 3/3)
CLIENT-SIDE SYSTEM LOGIC

Fig. 2-2 **Client-side Initialization**

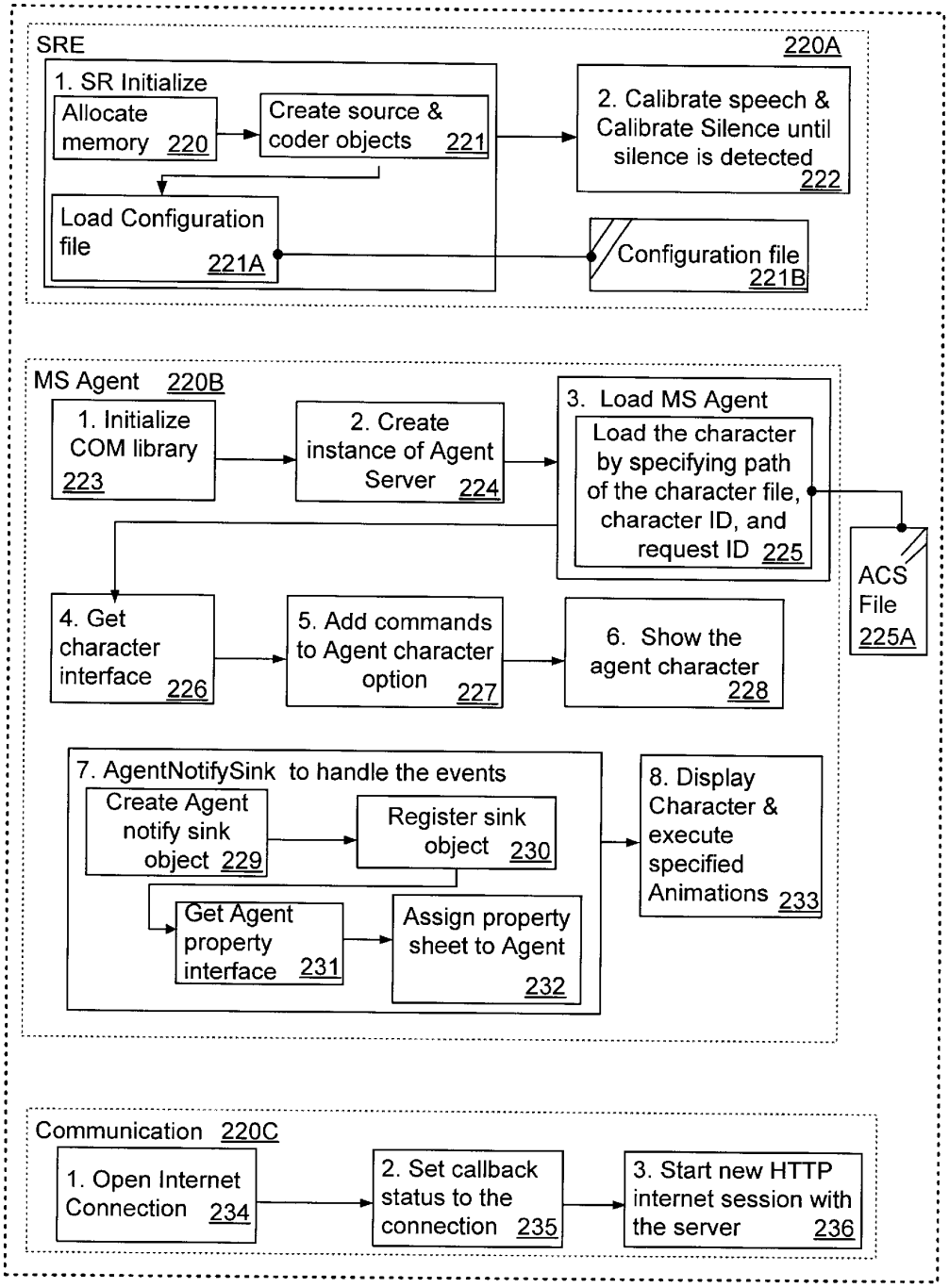


Fig. 3

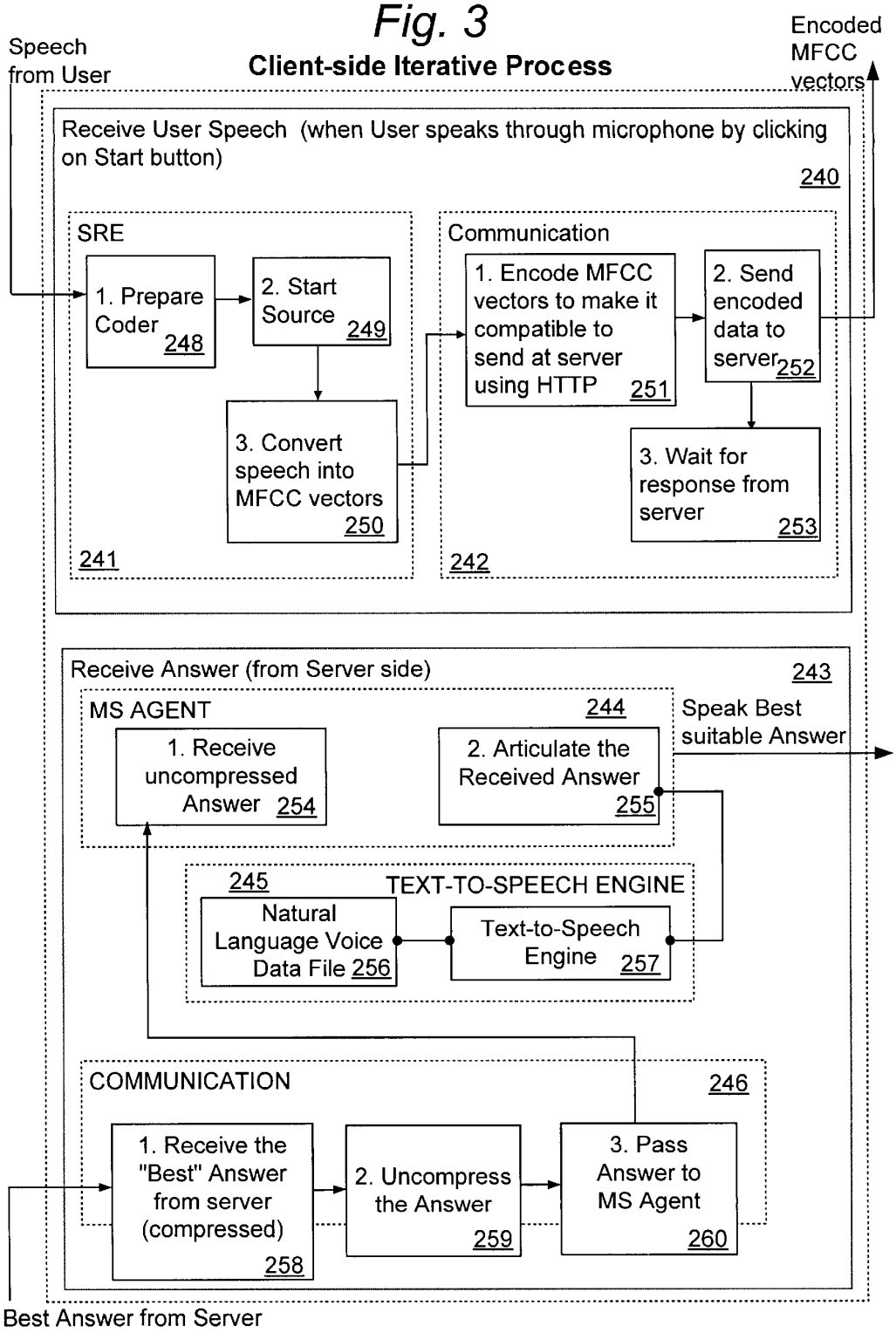

Client-side Iterative Process
Speech from User
Encoded MFCC vectors
Receive User Speech (when User speaks through microphone by clicking on Start button)
240
SRE
1. Prepare Coder 248
2. Start Source 249
3. Convert speech into MFCC vectors 250
241
Communication
1. Encode MFCC vectors to make it compatible to send at server using HTTP 251
2. Send encoded data to server 252
3. Wait for response from server 253
242
Receive Answer (from Server side)
243
MS AGENT
244
1. Receive uncompressed Answer 254
2. Articulate the Received Answer 255
Speak Best suitable Answer
245
TEXT-TO-SPEECH ENGINE
Natural Language Voice Data File 256
Text-to-Speech Engine 257
COMMUNICATION
246
1. Receive the "Best" Answer from server (compressed) 258
2. Uncompress the Answer 259
3. Pass Answer to MS Agent 260
Best Answer from Server

Fig. 4

Client-side Un-Initialization

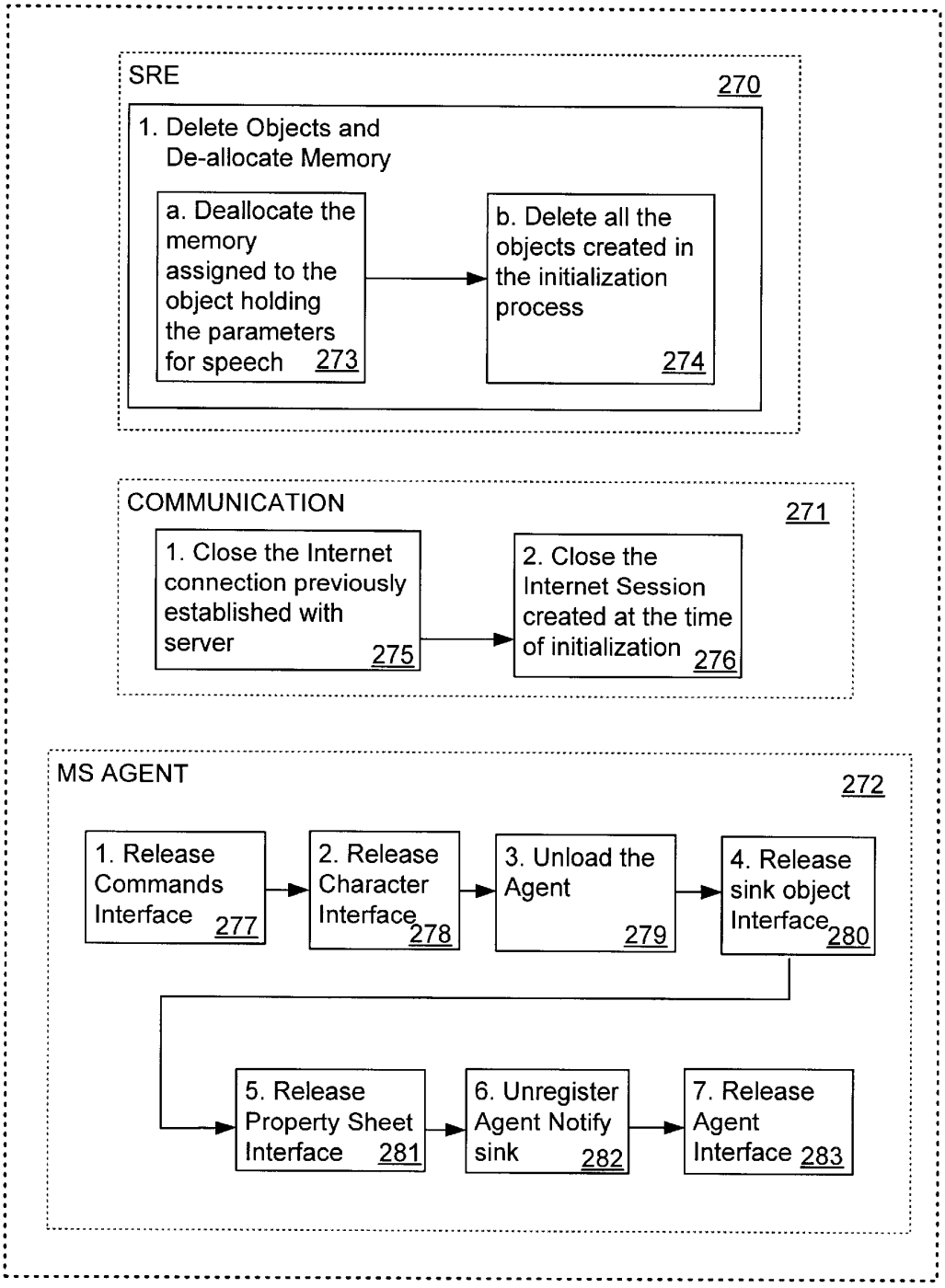
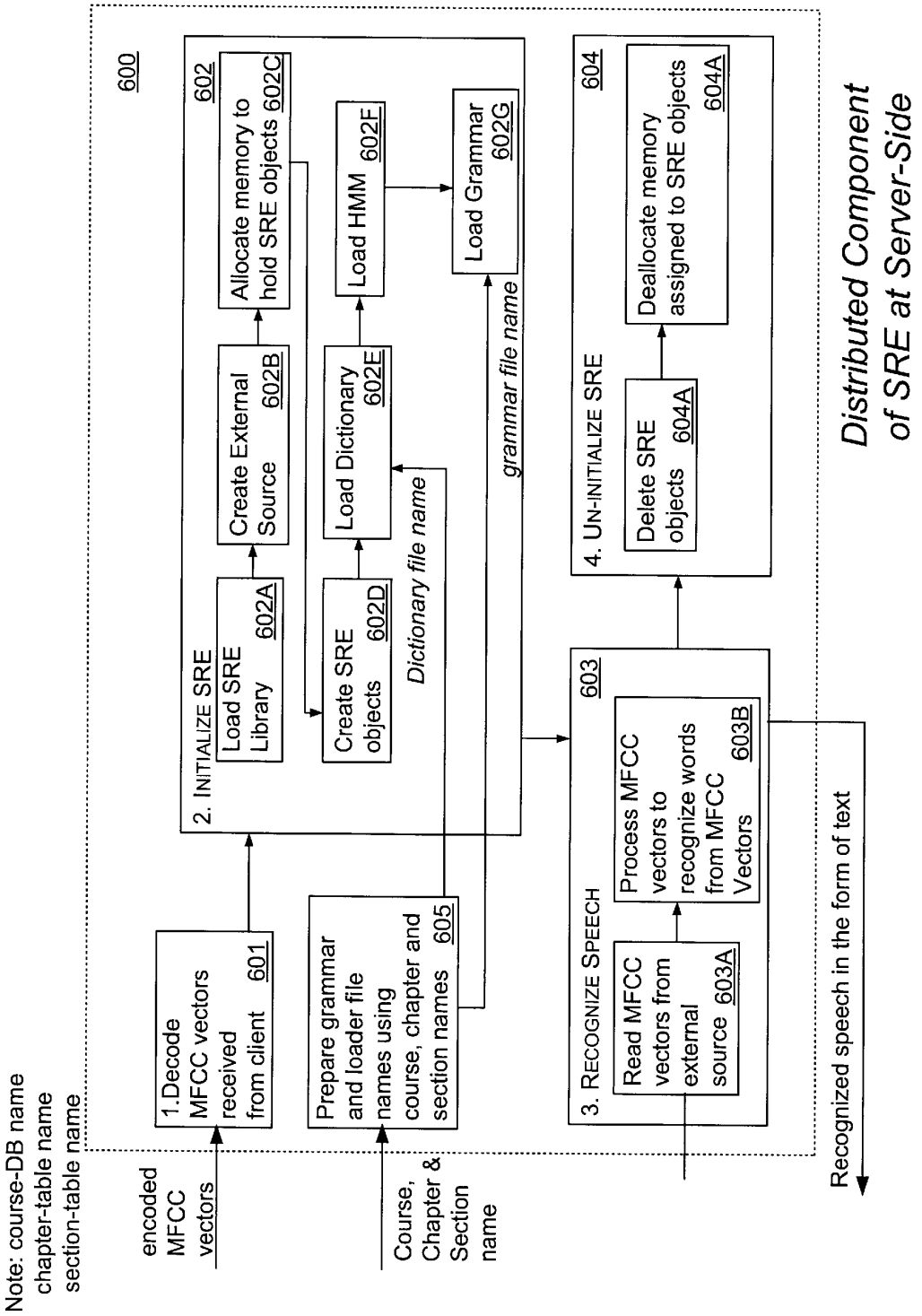


Fig. 4A


600
Distributed Component of SRE at Server-Side
Note: course-DB name
chapter-table name
section-table name
encoded MFCC vectors
Course, Chapter & Section name
Recognized speech in the form of text
1.Decode MFCC vectors received from client 601
Prepare grammar and loader file names using course, chapter and section names 605
2. INITIALIZE SRE 602
Load SRE Library 602A
Create External Source 602B
Allocate memory to hold SRE objects 602C
Create SRE objects 602D
Load Dictionary 602E
Load HMM 602F
Load Grammar 602G
Dictionary file name
grammar file name
3. RECOGNIZE SPEECH 603
Read MFCC vectors from external source 603A
Process MFCC vectors to recognize words from MFCC Vectors 603B
4. UN-INITIALIZE SRE 604
Delete SRE objects 604A
Deallocate memory assigned to SRE objects 604A

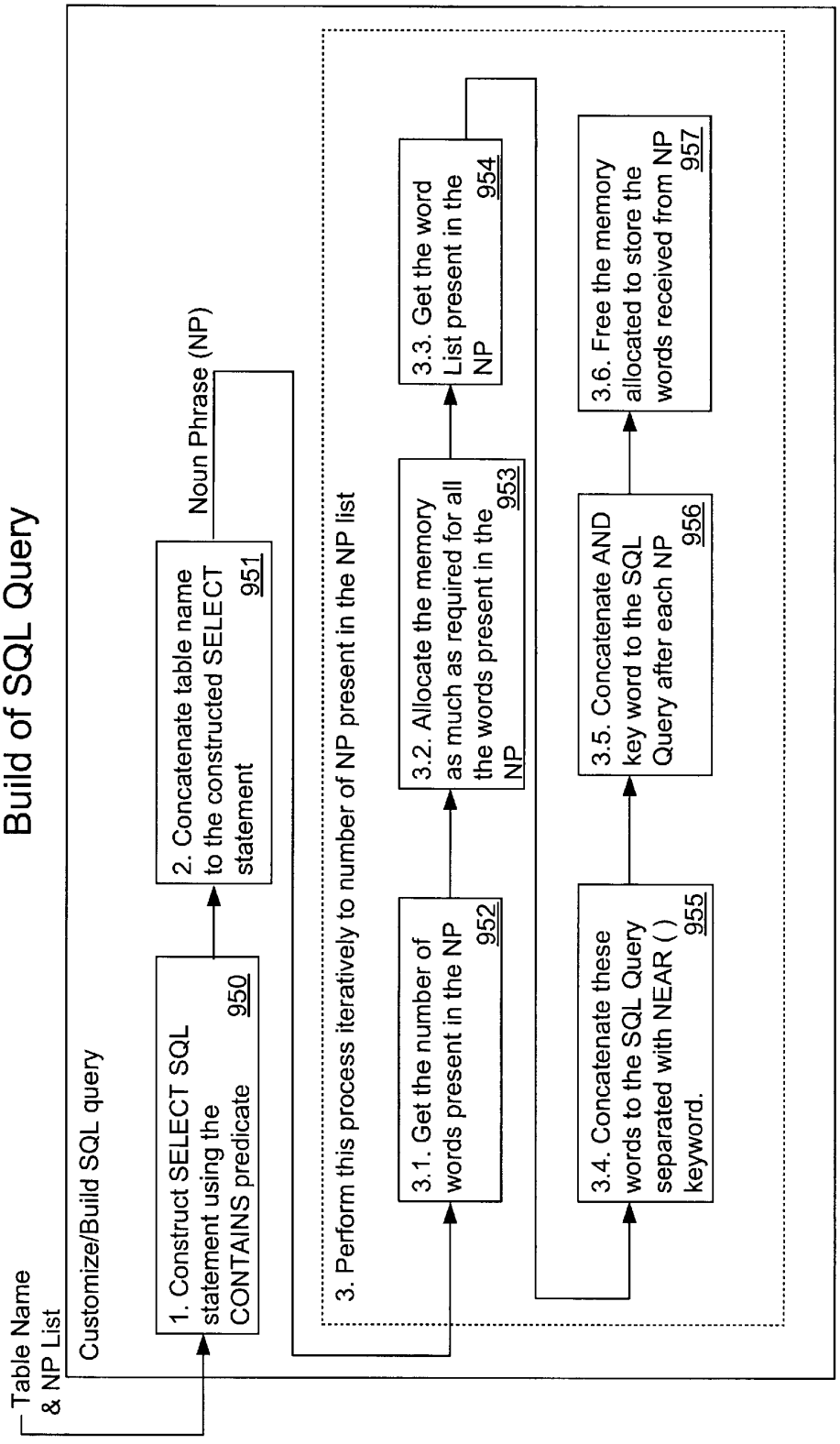


Fig. 4B

Build of SQL Query

Fig. 4C

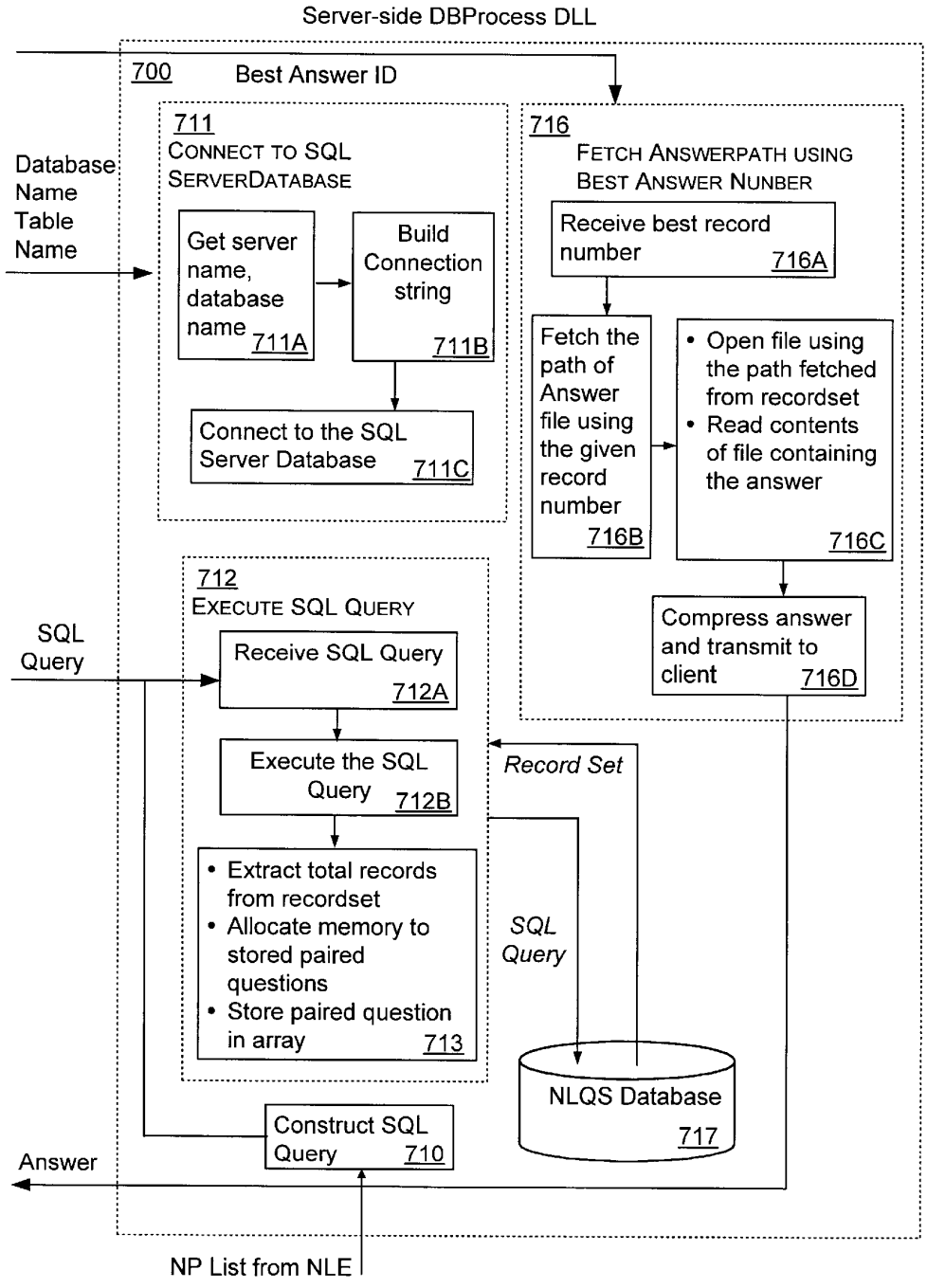

Server-side DBProcess DLL
700
Best Answer ID
711
CONNECT TO SQL SERVERDATABASE
Database Name
Table Name
Get server name, database name 711A
Build Connection string 711B
Connect to the SQL Server Database 711C
716
FETCH ANSWERPATH USING BEST ANSWER NUNBER
Receive best record number 716A
Fetch the path of Answer file using the given record number 716B
• Open file using the path fetched from recordset
• Read contents of file containing the answer 716C
Compress answer and transmit to client 716D
712
EXECUTE SQL QUERY
SQL Query
Receive SQL Query 712A
Execute the SQL Query 712B
Record Set
• Extract total records from recordset
• Allocate memory to stored paired questions
• Store paired question in array 713
SQL Query
NLQS Database 717
Construct SQL Query 710
Answer
NP List from NLE

Fig. 4D

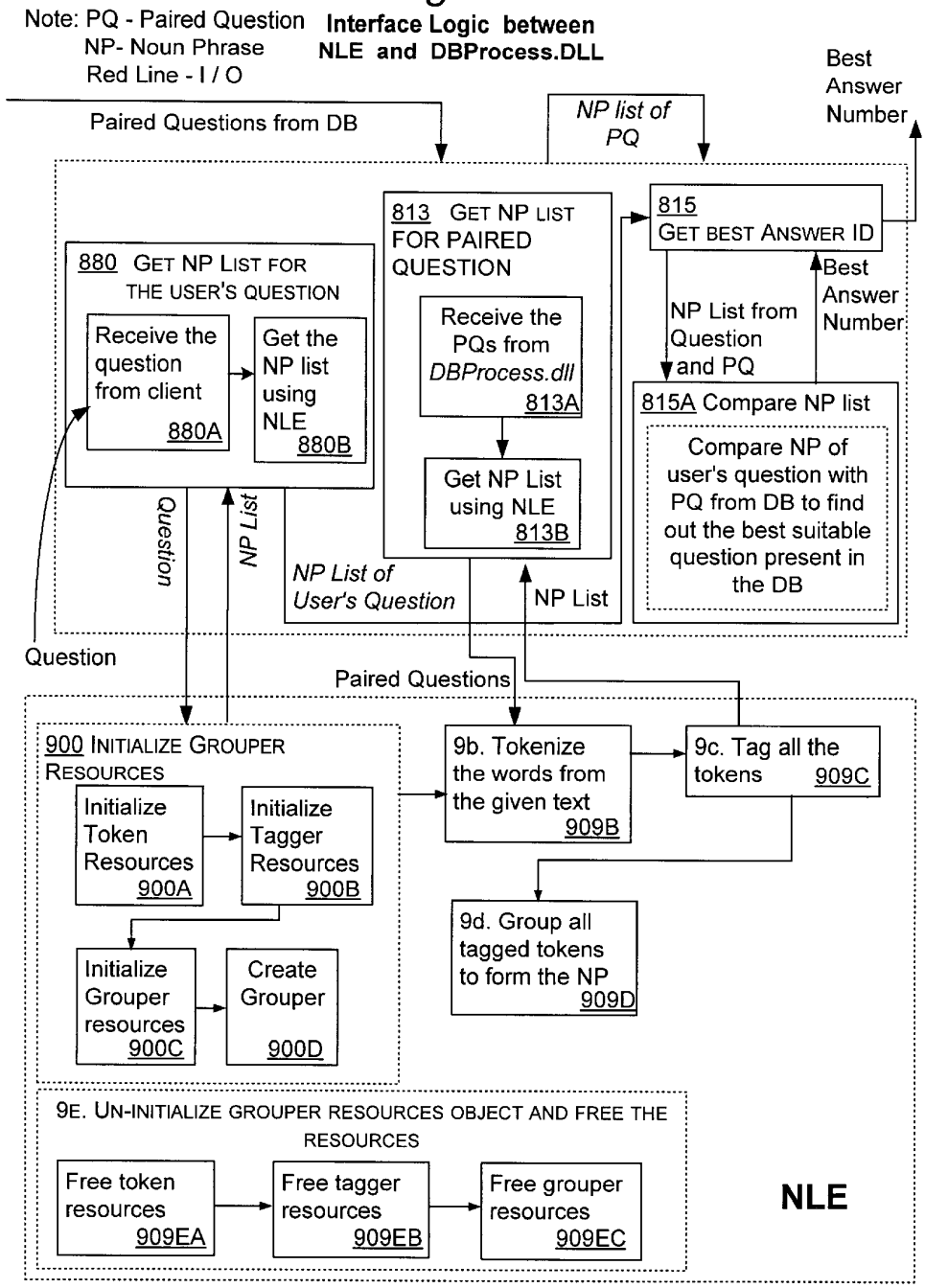

Note: PQ - Paired Question
NP- Noun Phrase
Red Line - I / O
Interface Logic between NLE and DBProcess.DLL
Paired Questions from DB
NP list of PQ
Best Answer Number
880 GET NP LIST FOR THE USER'S QUESTION
Receive the question from client 880A
Get the NP list using NLE 880B
813 GET NP LIST FOR PAIRED QUESTION
Receive the PQs from DBProcess.dll 813A
Get NP List using NLE 813B
815 GET BEST ANSWER ID
NP List from Question and PQ
Best Answer Number
815A Compare NP list
Compare NP of user's question with PQ from DB to find out the best suitable question present in the DB
Question
NP List
NP List of User's Question
NP List
Question
Paired Questions
900 INITIALIZE GROUPER RESOURCES
Initialize Token Resources 900A
Initialize Tagger Resources 900B
Initialize Grouper resources 900C
Create Grouper 900D
9b. Tokenize the words from the given text 909B
9c. Tag all the tokens 909C
9d. Group all tagged tokens to form the NP 909D
9E. UN-INITIALIZE GROUPER RESOURCES OBJECT AND FREE THE RESOURCES
Free token resources 909EA
Free tagger resources 909EB
Free grouper resources 909EC
NLE

Fig. 5

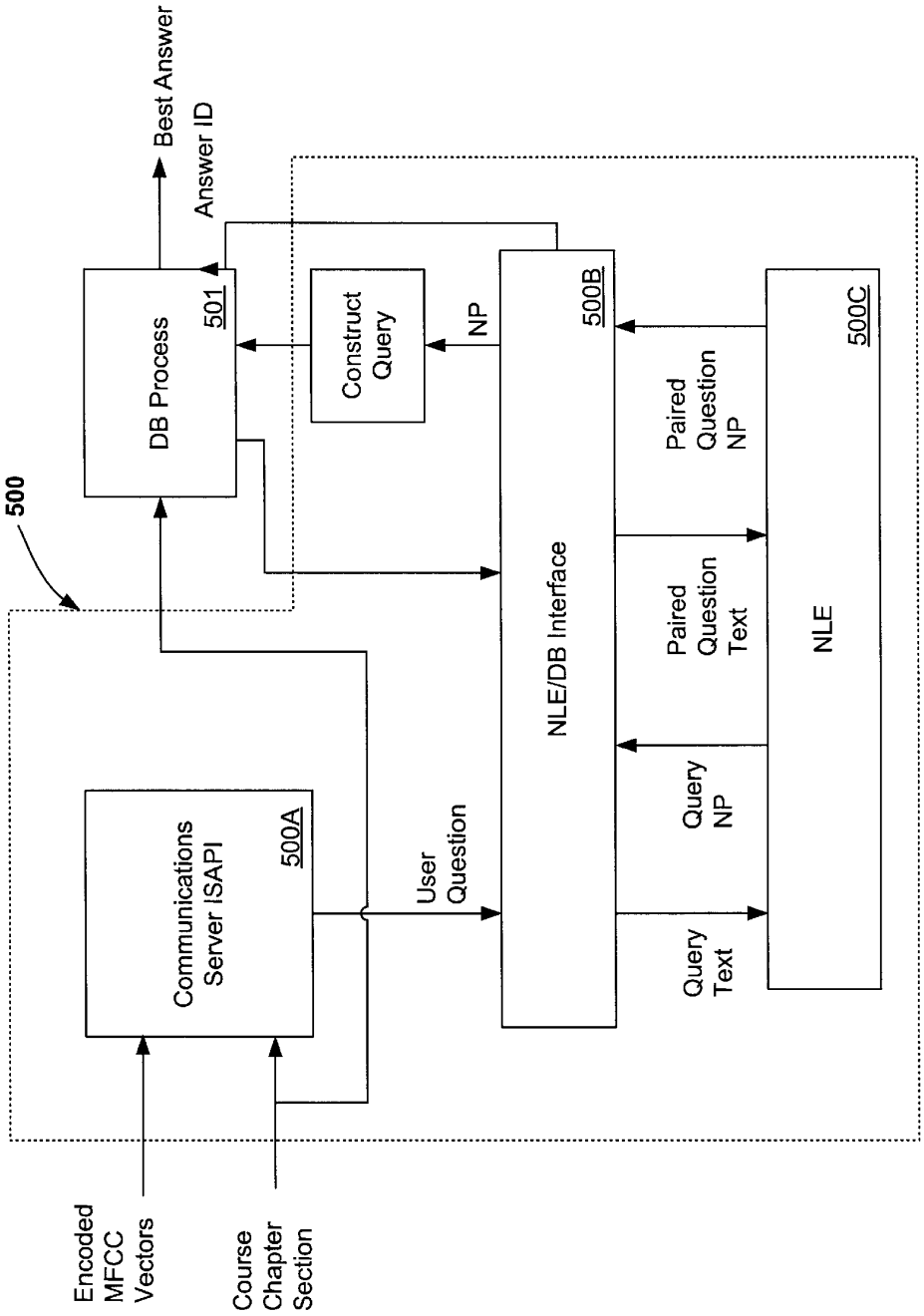

500
Best Answer
Answer ID
DB Process
501
Construct Query
NP
NLE/DB Interface
500B
Paired Question NP
Paired Question Text
Query NP
Query Text
NLE
500C
Communications Server ISAPI
500A
User Question
Encoded MFCC Vectors
Course Chapter Section

Fig.6

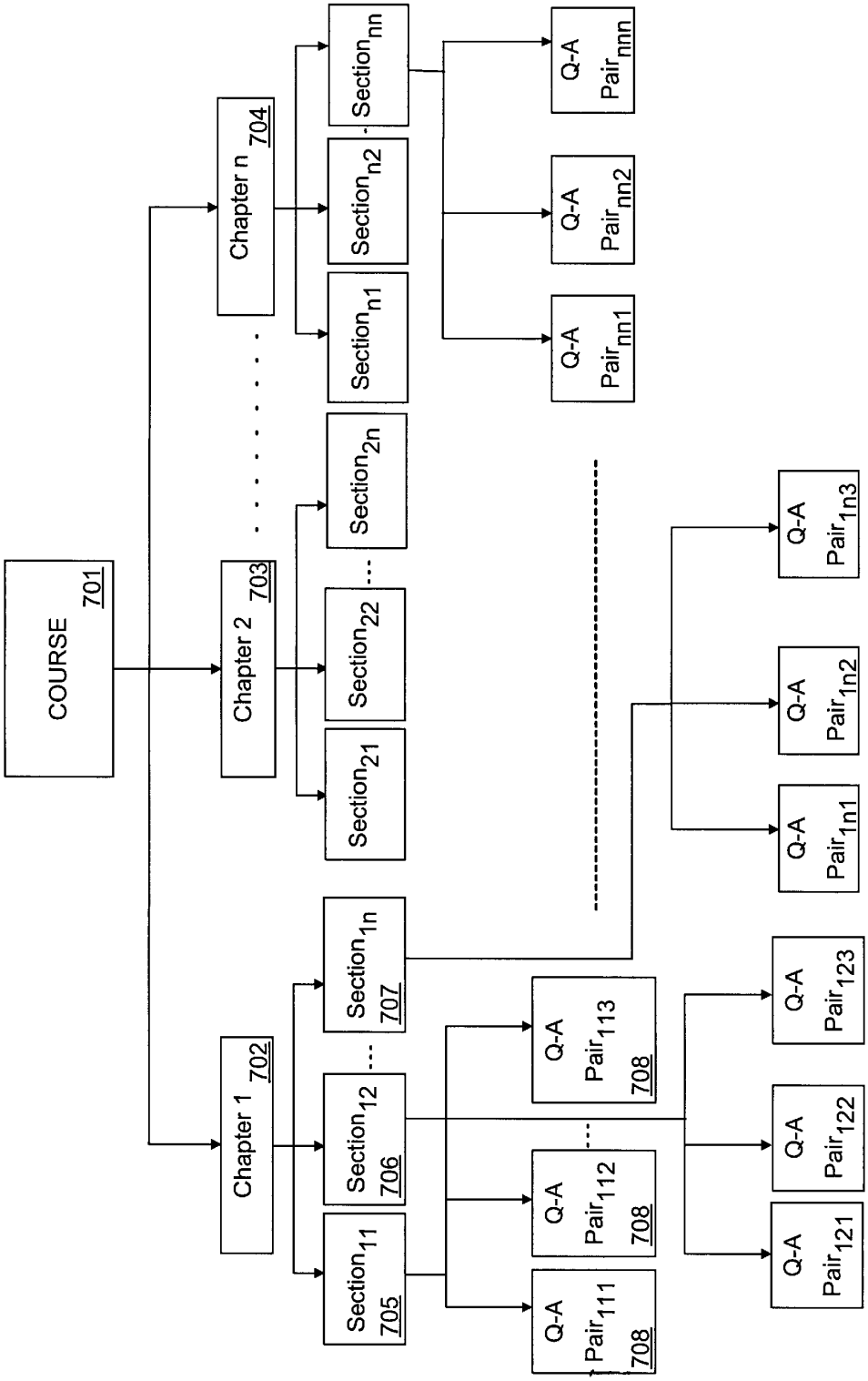

COURSE 701
Chapter 1 702
Chapter 2 703
Chapter n 704
Section11 705
Section12 706
Section1n 707
Section21
Section22
Section2n
Sectionn1
Sectionn2
Sectionnn
Q-A Pair111 708
Q-A Pair112 708
Q-A Pair113 708
Q-A Pair121
Q-A Pair122
Q-A Pair123
Q-A Pair1n1
Q-A Pair1n2
Q-A Pair1n3
Q-A Pairnn1
Q-A Pairnn2
Q-A Pairnnn

Fig. 7A

FIELD NAME 701A	DATA TYPE 702A	SIZE 703A	NULL 704A	PRIMARY KEY 705A	INDEXED? 706A
ChapterName 707A	Varchar	255	No	No	Yes
SectionName 708A	Varchar	255	No	No	Yes

Fig. 7B

FIELD NAME 720	DATA TYPE 721	SIZE 722	NULL 723	PRIMARY KEY 724	INDEXED? 725
Chapter_ID 726	Integer		No	Yes	Yes
Answer_ID 727	Char	5	No	UNIQUE	Yes
Section_Name 728	Varchar	255	No	UNIQUE	Yes
Answer_Title 729	Varchar	255	Yes	No	Yes
PairedQuestion 730	Text	16	No	No	Yes (Full-Text)
AnswerPath 731	Varchar	255	No	No	Yes
Creator 732	Varchar	50	No	No	Yes
Date_of_Creation 733	Date	-	No	No	Yes
Date_of_Modification 734	Date	-	No	No	Yes

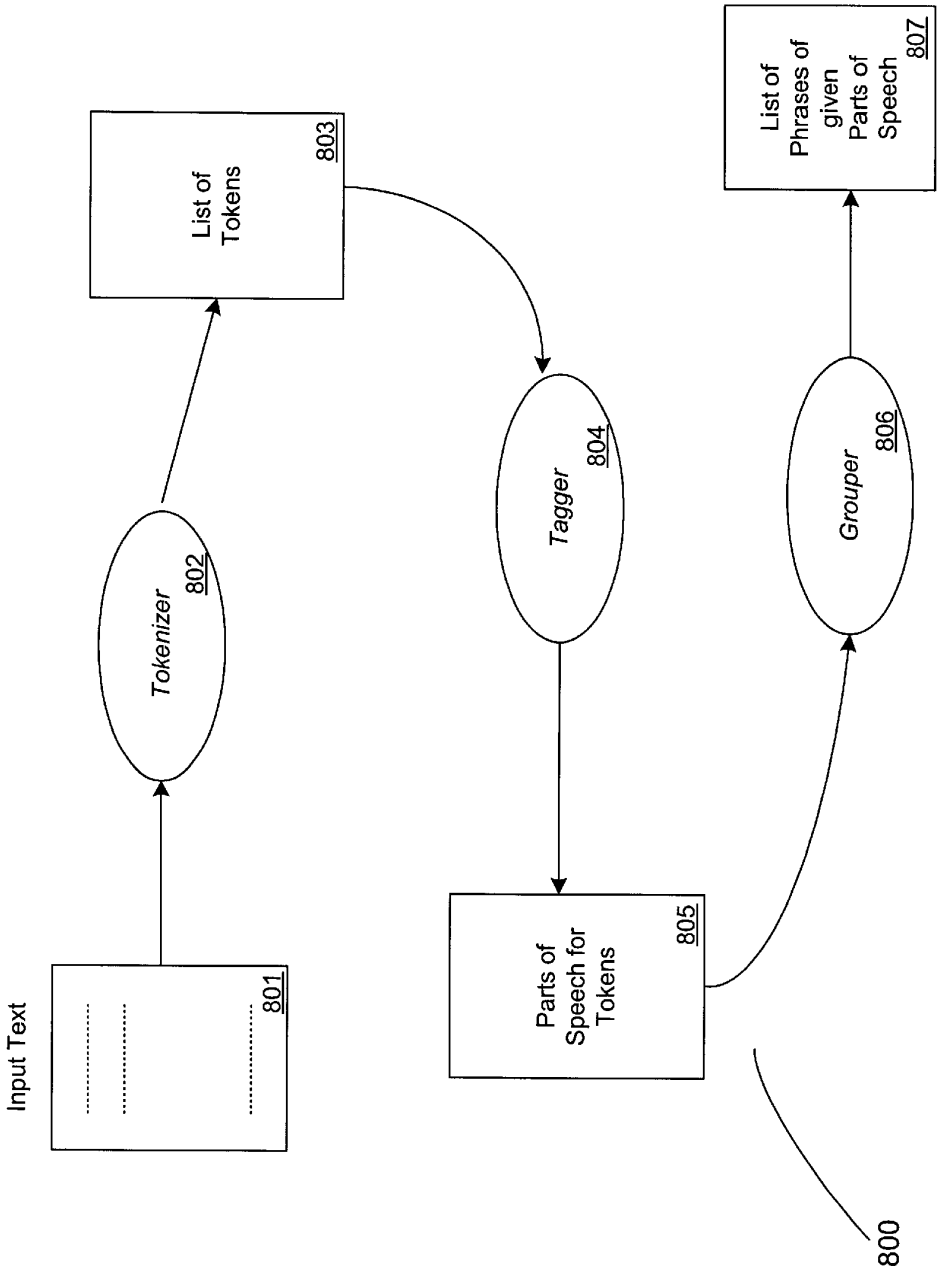
Fig. 7C

Field 720	Description 735
AnswerID 727	An integer - automatically incremented for user convenience
Section_Name 728	Name of section to which the particular record belongs. This field along with AnswerID has to be made primary key
Answer_Title 729	A short description of the answer
PairedQuestion 730	Contains one or more combinations of questions for the related answer whose path is stored in the next column AnswerPath
AnswerPath 731	Contains the path of text file, which contains theanswer to the rleated qusetions stored in the previous column
Creator 732	Name of content creator
Date_of_Creation 733	Date on which content has been added
Date_of_Modification 734	Date on which content has been changed or modified

Fig. 7D

FIELD 740	DATA TYPE 741	SIZE 742	NULL 743	PRIMARY KEY 744	INDEXED 745
Answer_ID 746	Char	5	No	Yes	Yes
Answer_Title 747	Varchar	255	Yes	No	No
PairedQuestion 748	Text	16	No	No	Yes (Full-Text)
Answer_Path 749	Varchar	255	No	No	No
Creator 750	Varchar	50	No	No	No
Date_of_Creation 751	Date	-	No	No	No
Date_of_Modification 752	Date	-	No	No	No

Fig. 8


Input Text
801
Tokenizer
802
List of Tokens
803
Tagger
804
Parts of Speech for Tokens
805
Grouper
806
List of Phrases of given Parts of Speech
807
800

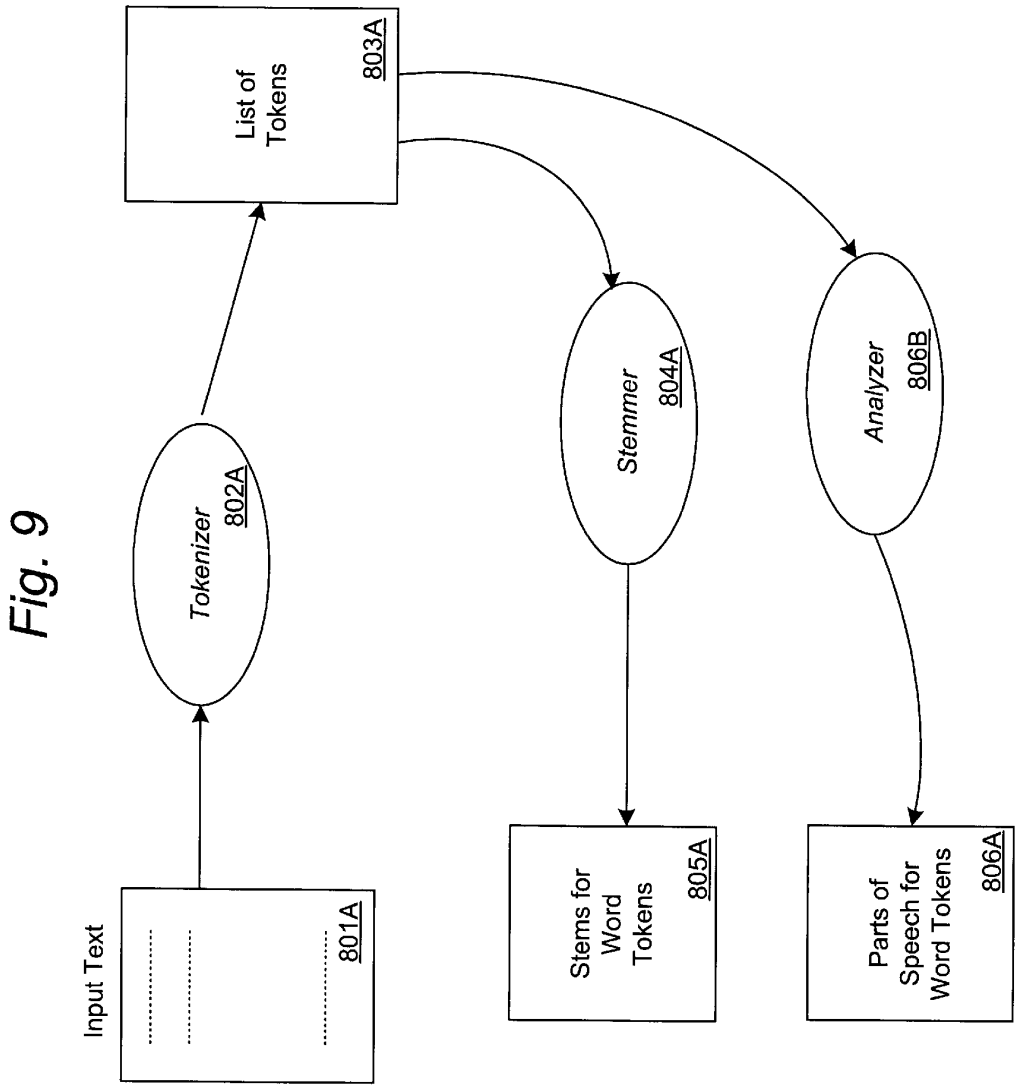

Input Text
801A
Tokenizer
802A
List of Tokens
803A
Stemmer
804A
Analyzer
806B
Stems for Word Tokens
805A
Parts of Speech for Word Tokens
806A

Fig. 9

Fig. 10

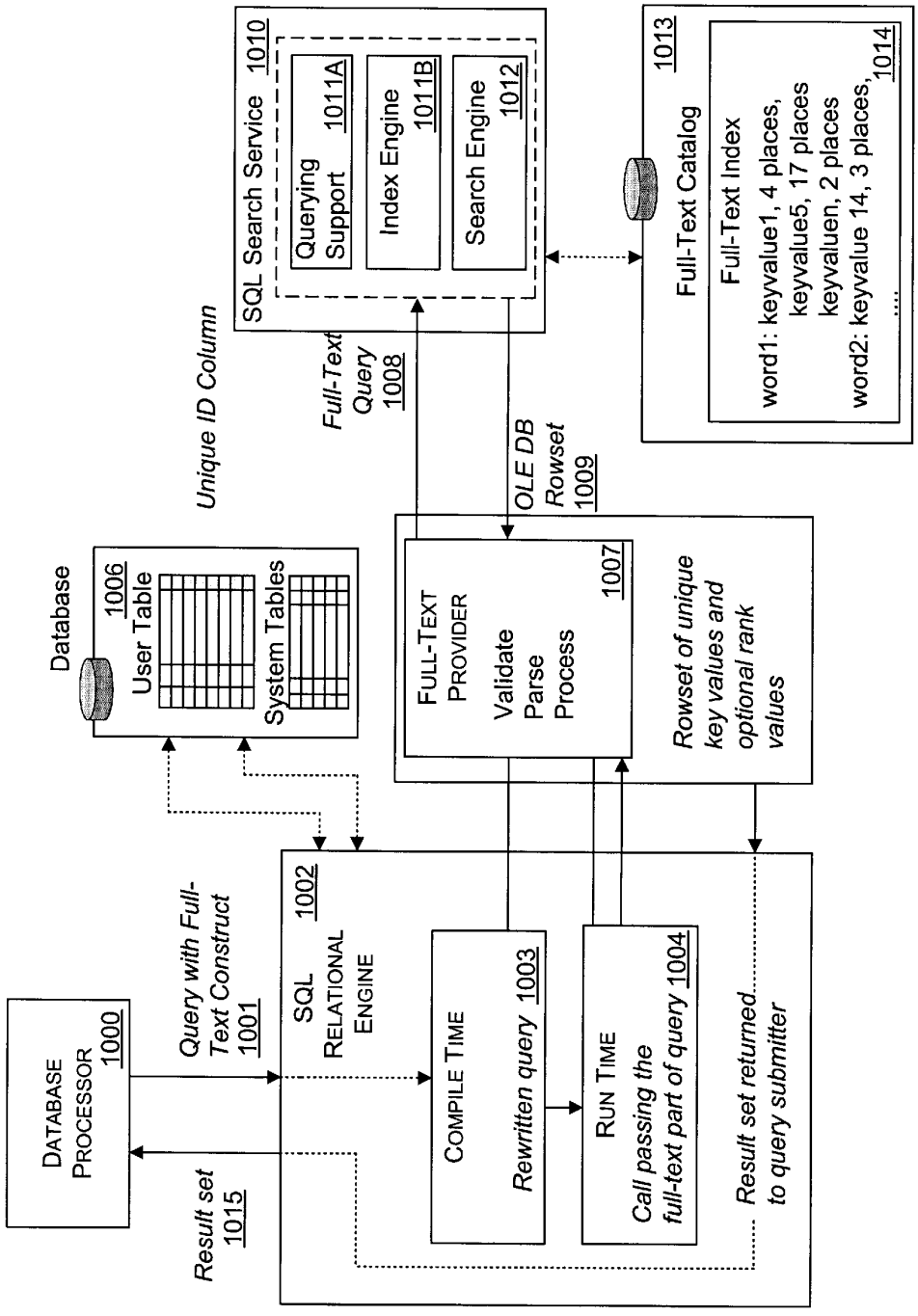

DATABASE PROCESSOR 1000
Query with Full-Text Construct 1001
Result set 1015
SQL RELATIONAL ENGINE 1002
COMPILE TIME
Rewritten query 1003
RUN TIME
Call passing the full-text part of query 1004
Result set returned to query submitter
Database
User Table 1006
System Tables
FULL-TEXT PROVIDER 1007
Validate
Parse
Process
Rowset of unique key values and optional rank values
OLE DB Rowset 1009
Full-Text Query 1008
Unique ID Column
SQL Search Service 1010
Querying Support 1011A
Index Engine 1011B
Search Engine 1012
Full-Text Catalog 1013
Full-Text Index
word1: keyvalue1, 4 places, keyvalue5, 17 places keyvaluen, 2 places
word2: keyvalue 14, 3 places, ... 1014

Fig. 11A

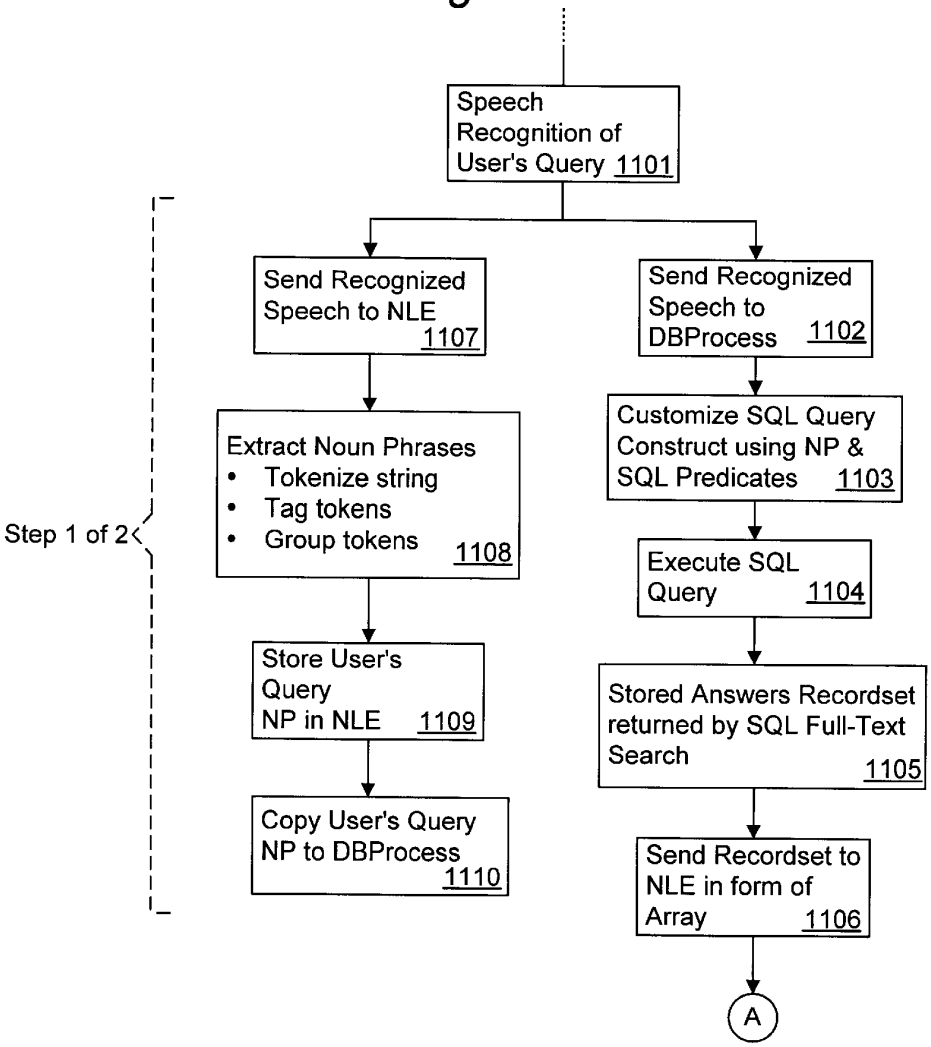

Speech Recognition of User's Query 1101
Send Recognized Speech to NLE 1107
Send Recognized Speech to DBProcess 1102
Extract Noun Phrases
• Tokenize string
• Tag tokens
• Group tokens 1108
Customize SQL Query Construct using NP & SQL Predicates 1103
Execute SQL Query 1104
Store User's Query NP in NLE 1109
Stored Answers Recordset returned by SQL Full-Text Search 1105
Copy User's Query NP to DBProcess 1110
Send Recordset to NLE in form of Array 1106
A
Step 1 of 2

Fig. 11B

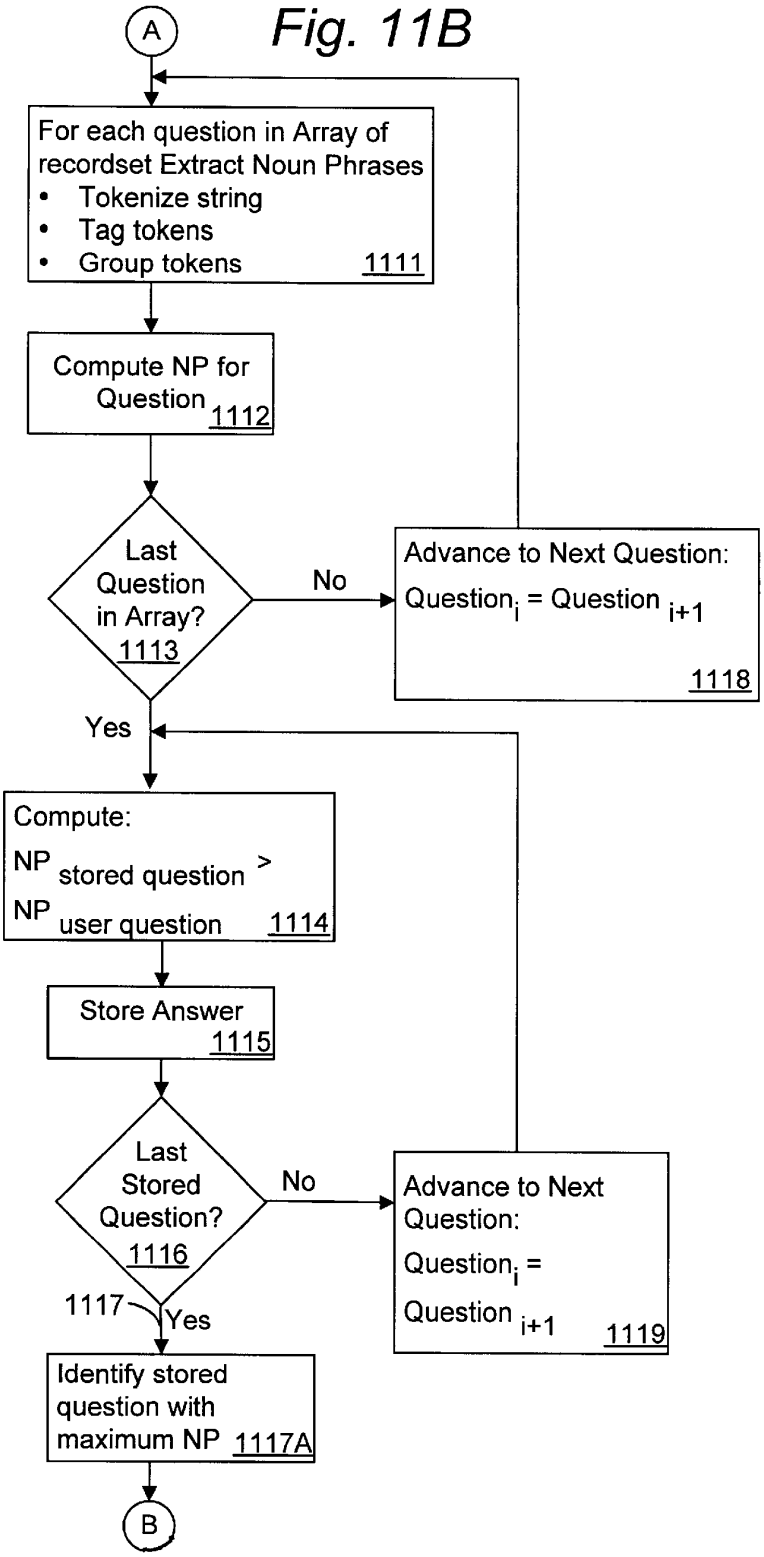

A
For each question in Array of recordset Extract Noun Phrases
• Tokenize string
• Tag tokens
• Group tokens
1111
Compute NP for Question
1112
Last Question in Array?
1113
No
Advance to Next Question:
Question i = Question i+1
1118
Yes
Compute:
NP stored question >
NP user question
1114
Store Answer
1115
Last Stored Question?
1116
No
Advance to Next Question:
Question i = Question i+1
1119
1117
Yes
Identify stored question with maximum NP
1117A
B

Fig. 11C

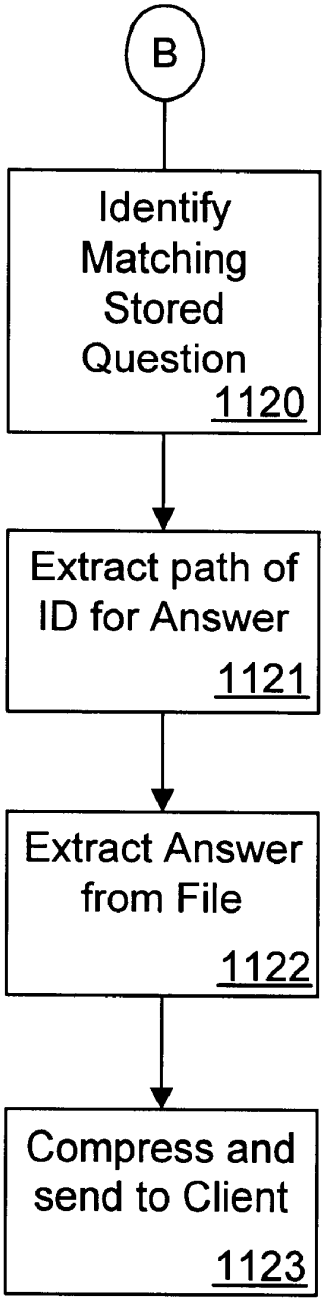

B
Identify Matching Stored Question
1120
Extract path of ID for Answer
1121
Extract Answer from File
1122
Compress and send to Client
1123

Fig. 12

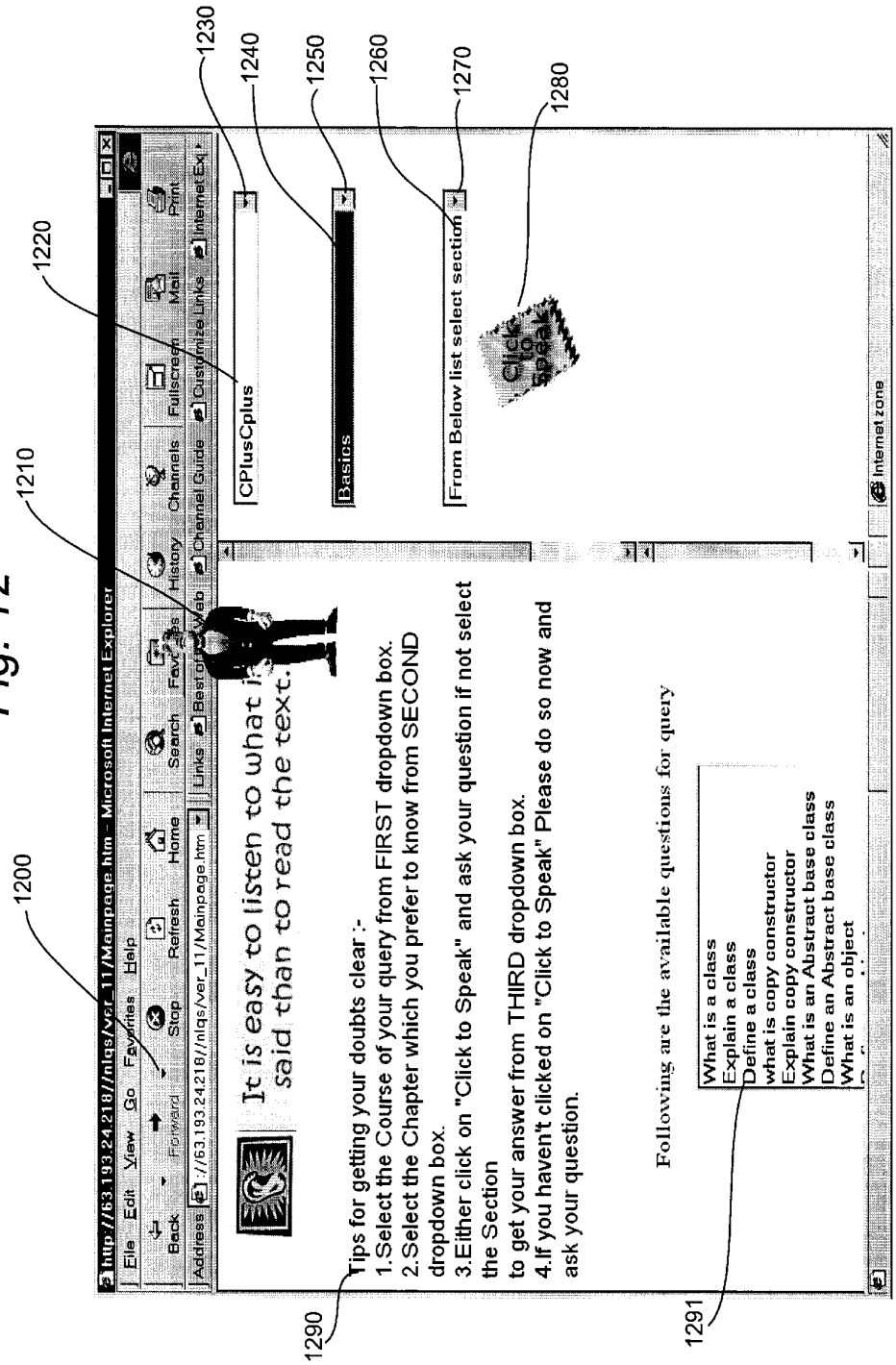

1200
1210
1220
1230
1240
1250
1260
1270
1280
1290
1291
CPlusCplus
Basics
From Below list select section
Click to Speak
It is easy to listen to what I
said than to read the text.
Tips for getting your doubts clear :-
1.Select the Course of your query from FIRST dropdown box.
2.Select the Chapter which you prefer to know from SECOND dropdown box.
3.Either click on "Click to Speak" and ask your question if not select the Section
to get your answer from THIRD dropdown box.
4.If you haven't clicked on "Click to Speak" Please do so now and ask your question.
Following are the available questions for query
What is a class
Explain a class
Define a class
what is copy constructor
Explain copy constructor
What is an Abstract base class
Define an Abstract base class
What is an object
Internet zone

Fig. 13

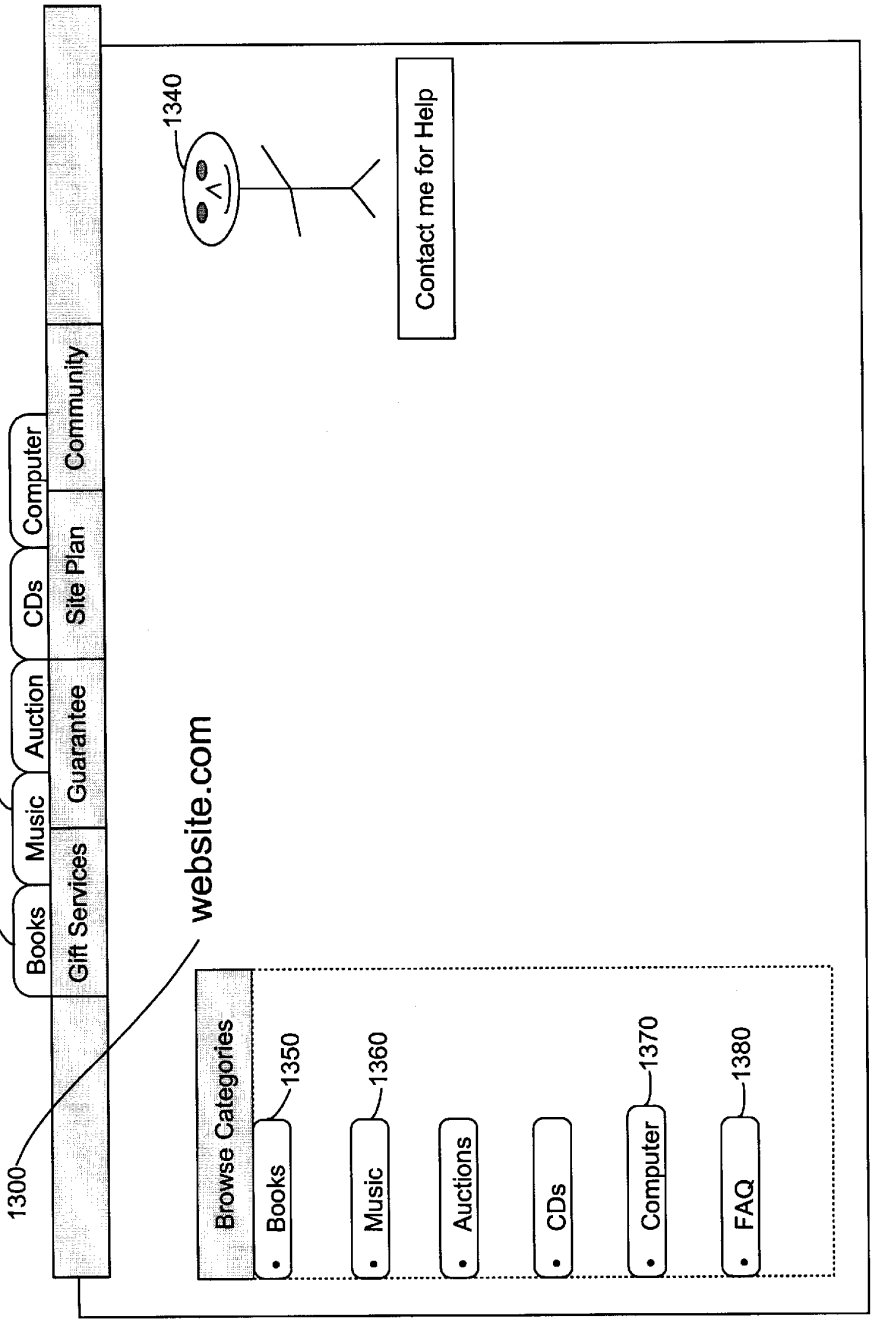

1330
Account Info
Help
1320
Books
Music
Auction
CDs
Computer
1310
Gift Services
Guarantee
Site Plan
Community
website.com
1300
1340
Contact me for Help
Browse Categories
1350
Books
1360
Music
Auctions
CDs
1370
Computer
1380
FAQ

Fig. 14

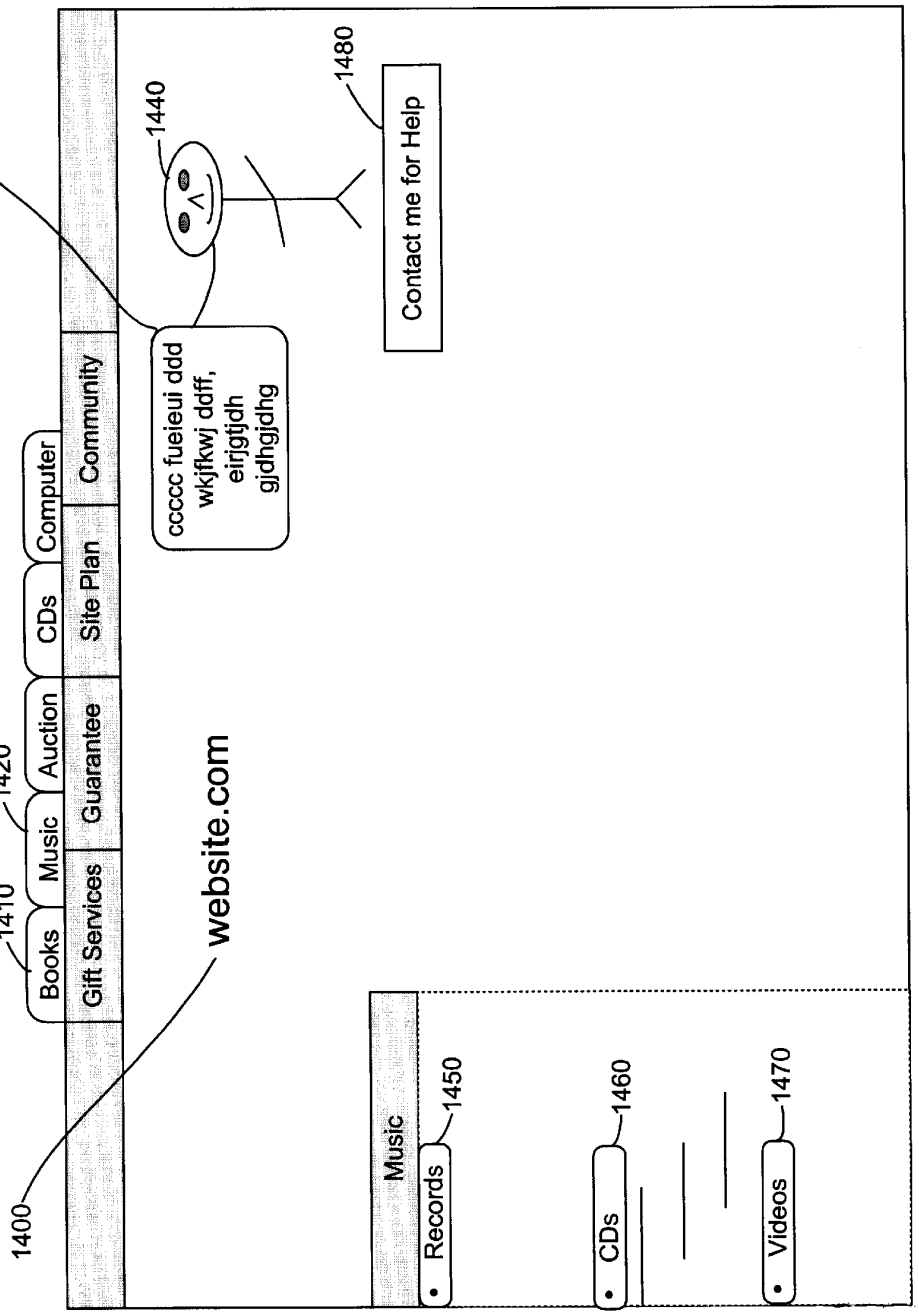

1400
website.com
1410
Books
Music
Gift Services
1420
Auction
Guarantee
CDs
Site Plan
Computer
Community
Account Info
Help
1430
1490
cccc fueieui ddd wkjfkwj ddff, eirjgtjdh gjdhgjdhg
1440
1480
Contact me for Help
Music
Records
1450
CDs
1460
Videos
1470

Fig. 15

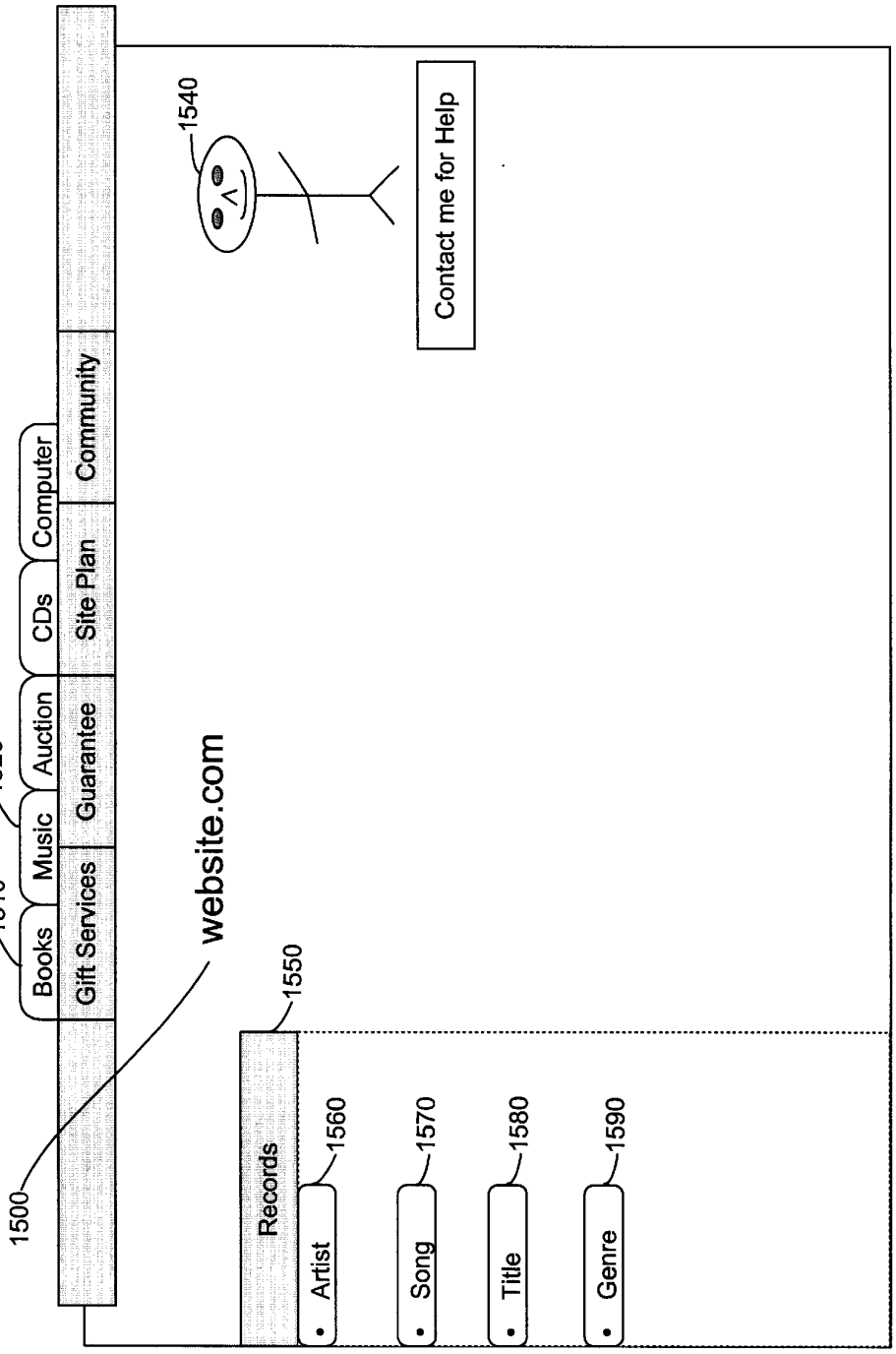

1500
website.com
1510
Books
Music
1520
Auction
Gift Services
Guarantee
CDs
Computer
Site Plan
Community
Account Info
Help
1530
1540
Contact me for Help
1550
Records
Artist
1560
Song
1570
Title
1580
Genre
1590

Fig. 16

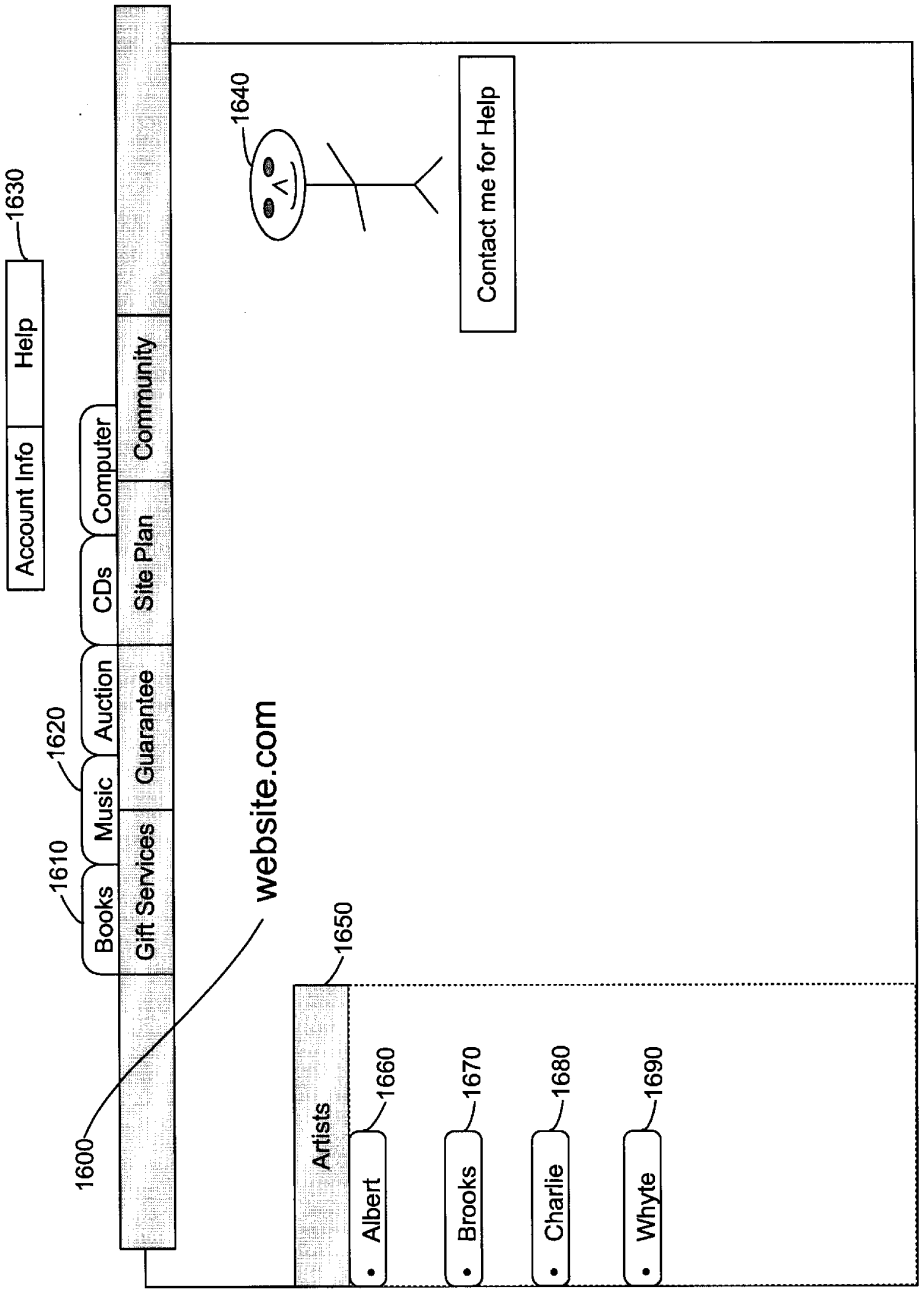

1600
website.com
1610
Books
Music
Gift Services
1620
Auction
Guarantee
CDs
Computer
Site Plan
Community
Account Info
Help
1630
1640
Contact me for Help
1650
Artists
1660
Albert
1670
Brooks
1680
Charlie
1690
Whyte

Fig. 17

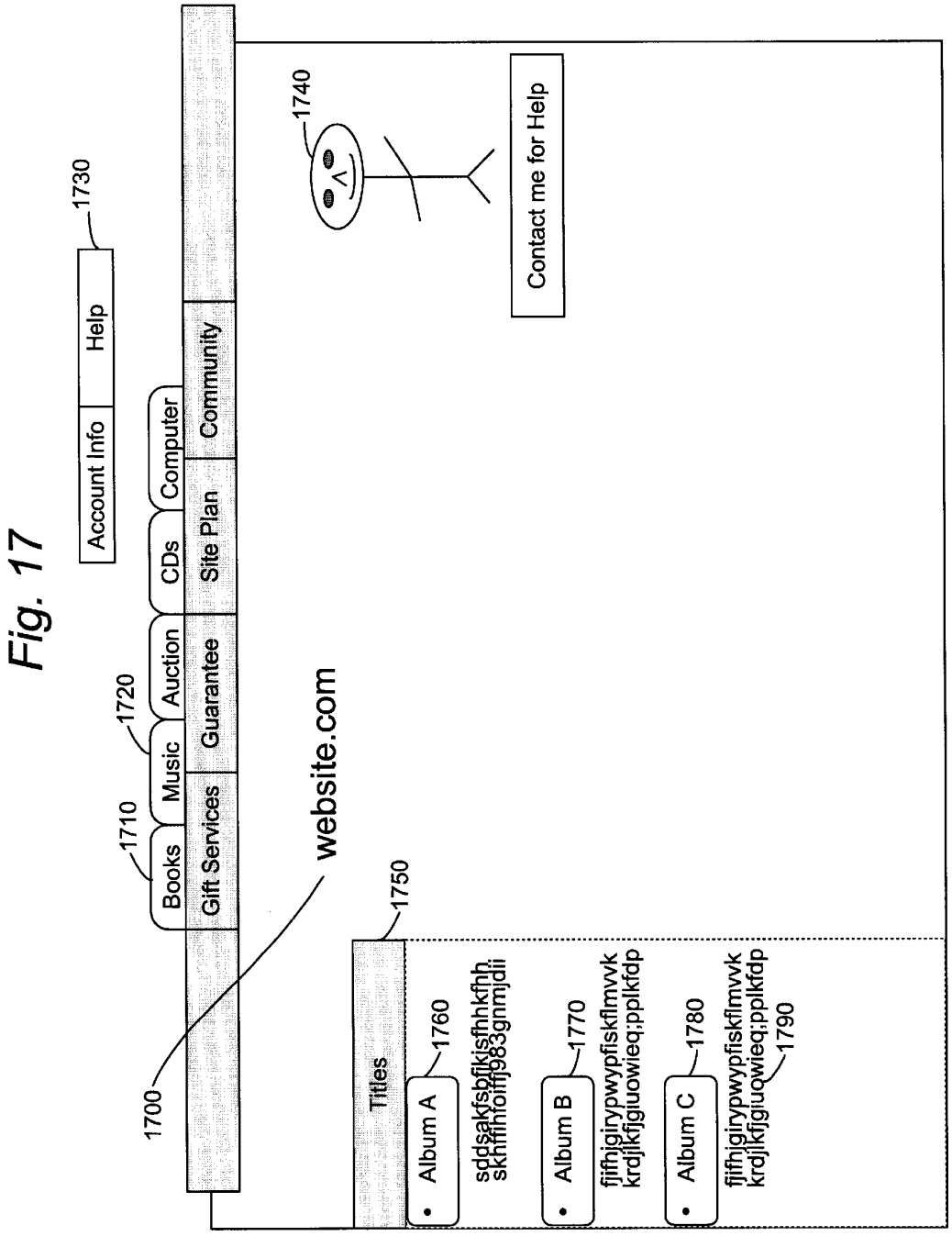

1730
Account Info
Help
1720
1710
Books
Music
Auction
CDs
Computer
Gift Services
Guarantee
Site Plan
Community
website.com
1700
1740
Contact me for Help
1750
Titles
Album A
1760
sddsakfsbfjkisfhhkfhh
skhfhfhoifjj983gnmjdii
Album B
1770
fjifhjgirypwypfiskflmvvk
krdjlkfjgiuowieq;pplkfdp
Album C
1780
fjifhjgirypwypfiskflmvvk
krdjlkfjgiuowieq;pplkfdp
1790

Fig. 18 (Page 1/2)

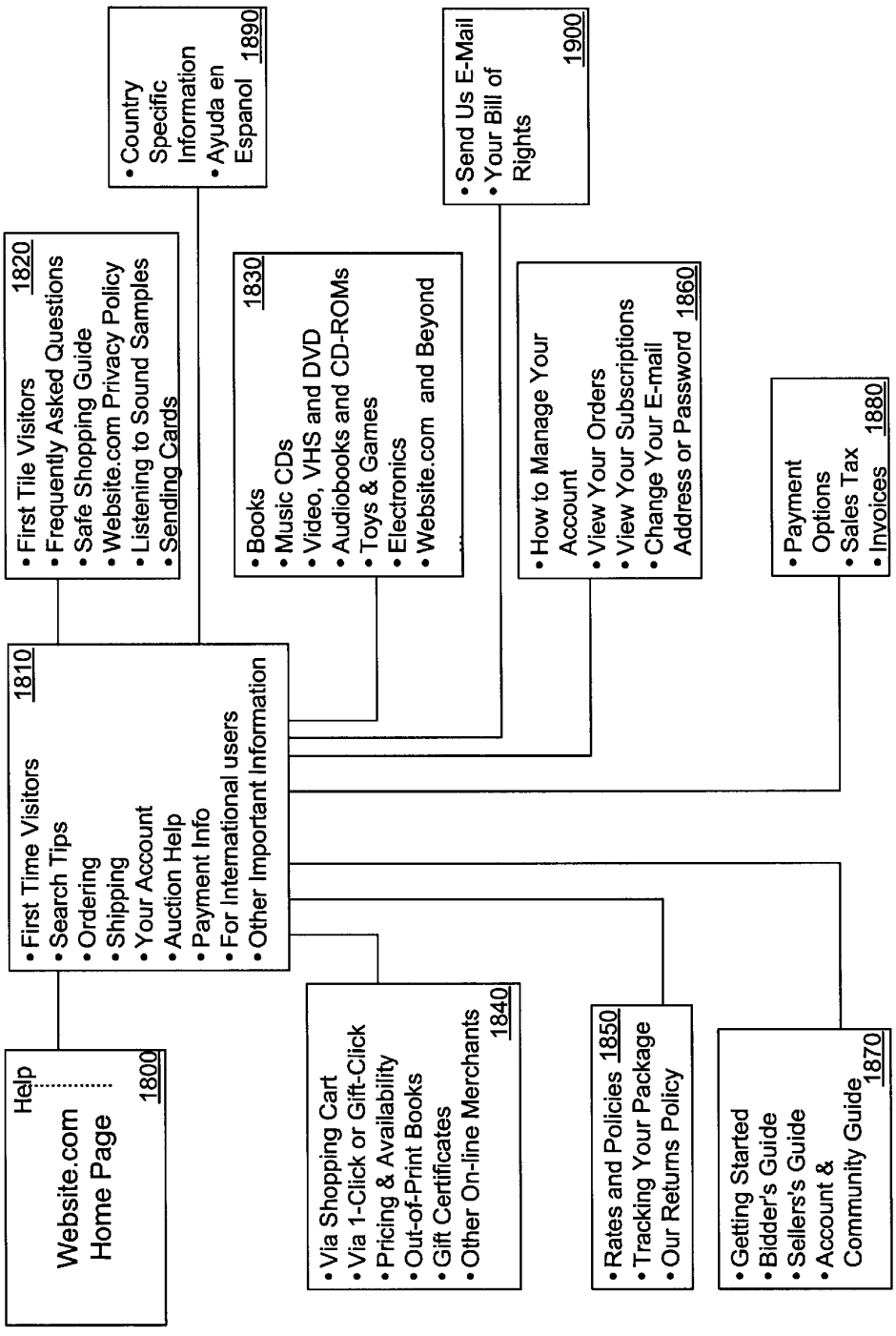
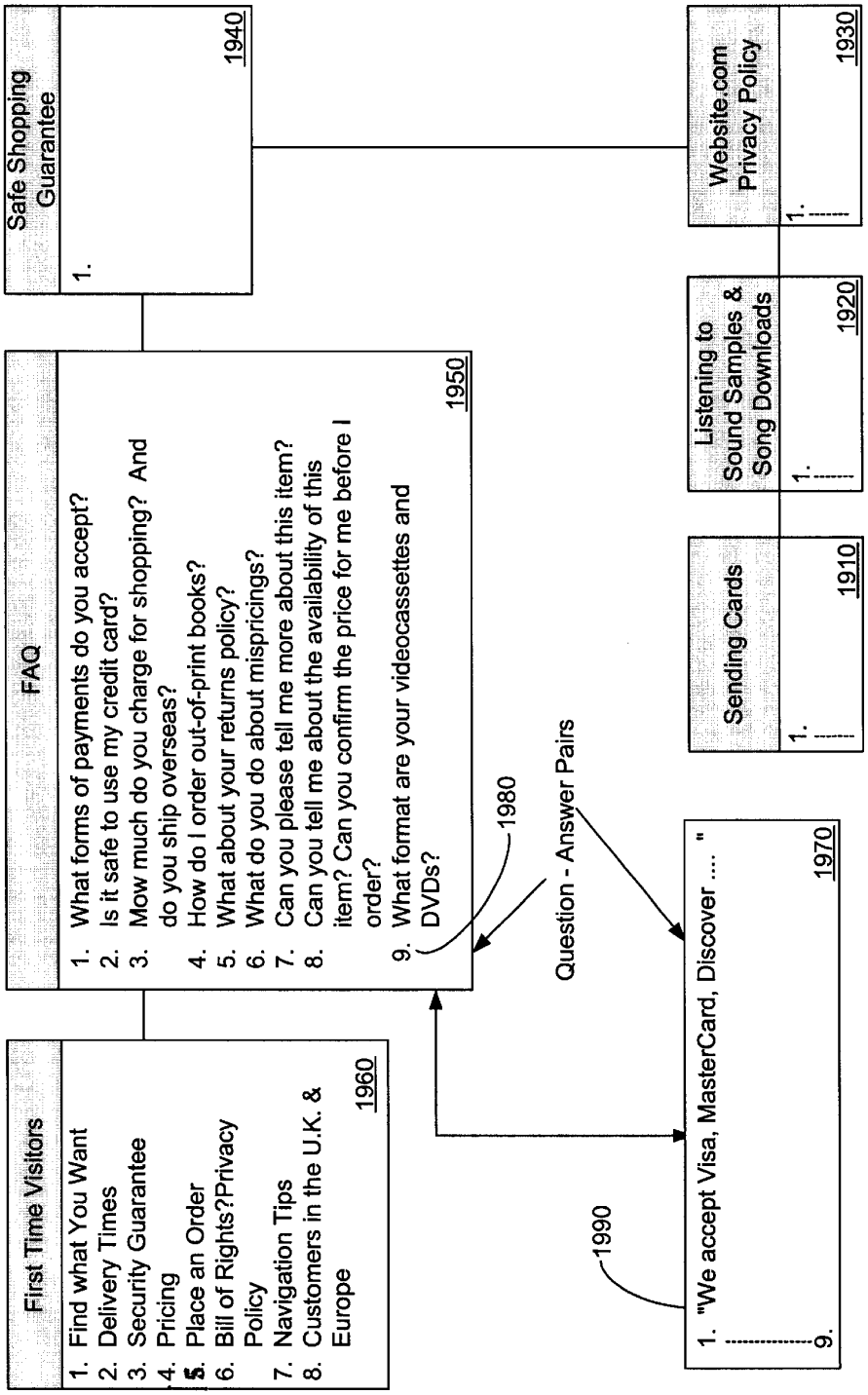


Fig. 18 (Page 2/2)



INTERACTIVE SPEECH BASED LEARNING/ TRAINING SYSTEM FORMULATING SEARCH QUERIES BASED ON NATURAL LANGUAGE PARSING OF RECOGNIZED USER QUERIES

RELATED APPLICATIONS

The present application is related to the following applications also filed contemporaneously herewith:

1) Ser. No. 09/439,145 entitled Distributed Real Time Speech Recognition System, attorney docket no. PHO 99-001;

2) Ser. No. 09/439,174 entitled Internet Server with Speech Support for Enhanced Interactivity—attorney docket no. PHO 99-003;

3) Ser. No. 09/439,060 entitled Intelligent Query Engine For Processing Voice Based Queries—attorney docket no. PHO 99-004;

The above are incorporated by reference herein.

FIELD OF THE INVENTION

The invention relates to a system and an interactive method for presenting an interactive, real-time speech-enabled tutorial over a distributed network such as the INTERNET or local intranet. This interactive system is especially useful when implemented over the World-Wide Web services (WWW of the INTERNET, functions so that a user/student can learn and interact with an animated agent who answers speech-based queries in a real-time fashion, thus providing a human-like dialog experience.

BACKGROUND OF THE INVENTION

The INTERNET, and in particular, the World-Wide Web (WWW, is growing in popularity and usage for both commercial and recreational purposes, and this trend is expected to continue. This phenomenon is being driven, in part, by the increasing and widespread use of personal computer systems and the availability of low cost INTERNET access.

The emergence of inexpensive INTERNET access devices and high speed access techniques such as ADSL, cable modems, satellite modems, and the like, are expected to further accelerate the mass usage of the WWW.

Accordingly, it is expected that the number of entities offering services, products, etc., over the WWW will increase dramatically over the coming years. Until now, however, the INTERNET "experience" for users has been limited mostly to non-voice based input/output devices, such as keyboards, intelligent electronic pads, mice, trackballs, printers, monitors, etc. This presents somewhat of a bottleneck for interacting over the WWW for a variety of reasons.

First, there is the issue of familiarity. Many kinds of applications lend themselves much more naturally and fluently to a voice-based environment. For instance, most people shopping for audio recordings are very comfortable with asking alive sales clerk in a record store for information on titles by a particular author, where they can be found in the store, etc. While it is often possible to browse and search on one's own to locate items of interest, it is usually easier and more efficient to get some form of human assistance first, and, with few exceptions, this request for assistance is presented in the form of a oral query. In addition, many persons cannot or will not, because of physical or psychological barriers, use any of the aforementioned conventional I/O devices. For example, many older persons cannot easily read the text presented on WWW pages, or understand the

layout/hierarchy of menus, or manipulate a mouse to make finely coordinated movements to indicate their selections. Many others are intimidated by the look and complexity of computer systems, WWW pages, etc., and therefore do not attempt to use online services for this reason as well.

Thus, applications which can mimic normal human interactions are likely to be preferred by potential on-line shoppers and persons looking for information over the WWW. It is also expected that the use of voice-based systems will increase the universe of persons willing to engage in e-commerce, e-learning, etc. To date, however, there are very few systems, if any, which permit this type of interaction, and, if they do, it is very limited. For example, various commercial programs sold by IBM (VIAVOICE™) and Kurzweil (DRAGON™) permit some user control of the interface (opening, closing files) and searching (by using previously trained URLs) but they do not present a flexible solution that can be used by a number of users across multiple cultures and without time consuming voice training. Typical prior efforts to implement voice based functionality in an INTERNET context can be seen in U.S. Pat. No. 5,819,220 incorporated by reference herein.

Another issue presented by the lack of voice-based systems is efficiency. Many companies are now offering technical support over the INTERNET, and some even offer live operator assistance for such queries. While this is very advantageous (for the reasons mentioned above) it is also extremely costly and inefficient, because a real person must be employed to handle such queries.

This presents a practical limit that results in long wait times for responses or high labor overheads. An example of this approach can be seen U.S. Pat. No. 5,802,526 also incorporated by reference herein. In general, a service presented over the WWW is far more desirable if it is "scalable," or, in other words, able to handle an increasing amount of user traffic with little if any perceived delay or troubles by a prospective user.

In a similar context, while remote learning has become an increasingly popular option for many students, it is practically impossible for an instructor to be able to field questions from more than one person at a time. Even then, such interaction usually takes place for only a limited period of time because of other instructor time constraints. To date, however, there is no practical way for students to continue a human-like question and answer type dialog after the learning session is over, or without the presence of the instructor to personally address such queries.

Conversely, another aspect of emulating a human-like dialog involves the use of oral feedback. In other words, many persons prefer to receive answers and information in audible form. While a form of this functionality is used by some websites to communicate information to visitors, it is not performed in a real-time, interactive question-answer dialog fashion so its effectiveness and usefulness is limited.

Yet another area that could benefit from speech-based interaction involves so-called "search" engines used by INTERNET users to locate information of interest at web sites, such as the those available at YAHOO®.com, METACRAWLER®.com, EXCITE®.com, etc. These tools permit the user to form a search query using either combinations of keywords or metacategories to search through a web page database containing text indices associated with one or more distinct web pages. After processing the user's request, therefore, the search engine returns a number of hits which correspond, generally, to URL pointers and text excerpts from the web pages that represent the closest match

made by such search engine for the particular user query based on the search processing logic used by search engine. The structure and operation of such prior art search engines, including the mechanism by which they build the web page database, and parse the search query, are well known in the art. To date, applicant is unaware of any such search engine that can easily and reliably search and retrieve information based on speech input from a user.

There are a number of reasons why the above environments (e-commerce, e-support, remote learning, INTERNET searching, etc.) do not utilize speech-based interfaces, despite the many benefits that would otherwise flow from such capability. First, there is obviously a requirement that the output of the speech recognizer be as accurate as possible. One of the more reliable approaches to speech recognition used at this time is based on the Hidden Markov Model (HMN)—a model used to mathematically describe any time series. A conventional usage of this technique is disclosed, for example, in U.S. Pat. No. 4,587,670 incorporated by reference herein. Because speech is considered to have an underlying sequence of one or more symbols, the HMM models corresponding to each symbol are trained on vectors from the speech waveforms. The Hidden Markov Model is a finite set of states, each of which is associated with a (generally multi-dimensional) probability distribution. Transitions among the states are governed by a set of probabilities called transition probabilities. In a particular state an outcome or observation can be generated, according to the associated probability distribution. This finite state machine changes state once every time unit, and each time t such that a state j is entered, a spectral parameter vector O_t is generated with probability density $B_j(O_t)$. It is only the outcome, not the state visible to an external observer and therefore states are "hidden" to the outside; hence the name Hidden Markov Model. The basic theory of HMMs was published in a series of classic papers by Baum and his colleagues in the late 1960's and early 1970's. HMMs were first used in speech applications by Baker at Carnegie Mellon, by Jelenik and colleagues at IBM in the late 1970's and by Steve Young and colleagues at Cambridge University, UK in the 1990's. Some typical papers and texts are as follows:

1. L. E. Baum, T. Petrie, "Statistical inference for probabilistic functions for finite state Markov chains", Ann. Math. Stat., 37:1554–1563,1966
2. L. E. Baum, "An inequality and associated maximation technique in statistical estimation for probabilistic functions of Markov processes", Inequalities 3: 1–8, 1972
3. J. H. Baker, "The dragon system—An Overview", IEEE Trans. on ASSP Proc., ASSP-23(1): 24–29, February, 1975
4. F. Jeninek et al, "Continuous Speech Recognition: Statistical methods" in Handbook of Statistics, II, P. R. Kristnaiad, Ed. Amsterdam, The Netherlands, North-Holland, 1982
5. L. R. Bahl, F. Jeninek, R. L. Mercer, "A maximum likelihood approach to continuous speech recognition", IEEE Trans. Pattern Anal. Mach. Intell., PAMI-5: 179–190,1983
6. J. D. Ferguson, "Hidden Markov Analysis: An Introduction", in Hidden Markov Models for Speech, Institute of Defense Analyses, Princeton, N.J. 1980.
7. H. R. Rabiner and B. H. Juang, "Fundamentals of Speech Recognition", Prentice Hall, 1993
8. H. R. Rabiner, "Digital Processing of Speech Signals", Prentice Hall, 1978

More recently research has progressed in extending HMM and combining HMMs with neural networks to speech recognition applications at various laboratories. The following is a representative paper:

9. Nelson Morgan, Hervé Bourlard, Steve Renals, Michael Cohen and Horacio Franco (1993), Hybrid Neural Network/Hidden Markov Model Systems for Continuous Speech Recognition. *Journal of Pattern Recognition and Artificial Intelligence*, Vol. 7, No. 4 pp. 899–916.

Also in I. Guyon and P. Wang editors, *Advances in Pattern Recognition Systems using Neural Networks*, Vol. 7 of a Series in Machine Perception and Artificial Intelligence. World Scientific, February 1994.

All of the above are hereby incorporated by reference. While the HMM-based speech recognition yields very good results, contemporary variations of this technique cannot guarantee a word accuracy requirement of 100% exactly and consistently, as will be required for WWW applications for all possible all user and environment conditions. Thus, although speech recognition technology has been available for several years, and has improved significantly, the technical requirements have placed severe restrictions on the specifications for the speech recognition accuracy that is required for an application that combines speech recognition and natural language processing to work satisfactorily.

In contrast to word recognition, Natural language processing (NLP) is concerned with the parsing, understanding and indexing of transcribed utterances and larger linguistic units. Because spontaneous speech contains many surface phenomena such as disfluencies,—hesitations, repairs and restarts, discourse markers such as 'well' and other elements which cannot be handled by the typical speech recognizer, it is the problem and the source of the large gap that separates speech recognition and natural language processing technologies. Except for silence between utterances, another problem is the absence of any marked punctuation available for segmenting the speech input into meaningful units such as utterances. For optimal NLP performance, these types of phenomena should be annotated at its input. However, most continuous speech recognition systems produce only a raw sequence of words. Examples of conventional systems using NLP are shown in U.S. Pat. Nos. 4,991,094, 5,068,789, 5,146,405 and 5,680,628, all of which are incorporated by reference herein.

Second, most of the very reliable voice recognition systems are speaker-dependent, requiring that the interface be "trained" with the user's voice, which takes a lot of time, and is thus very undesirable from the perspective of a WWW environment, where a user may interact only a few times with a particular website. Furthermore, speaker-dependent systems usually require a large user dictionary (one for each unique user) which reduces the speed of recognition. This makes it much harder to implement a real-time dialog interface with satisfactory response capability (i.e., something that mirrors normal conversation—on the order of 3–5 seconds is probably ideal). At present, the typical shrink-wrapped speech recognition application software include offerings from IBM (VIAVOICE™) and Dragon Systems (DRAGON™). While most of these applications are adequate for dictation and other transcribing applications, they are woefully inadequate for applications such as NLQS where the word error rate must be close to 0%. In addition these offerings require long training times and are typically are non client-server configurations. Other types of trained systems are discussed in U.S. Pat. No. 5,231,670 assigned to Kurzweil, and which is also incorporated by reference herein.

Another significant problem faced in a distributed voice-based system is a lack of uniformity/control in the speech recognition process. In a typical stand-alone implementation of a speech recognition system, the entire SR engine runs on a single client. A well-known system of this type is depicted in U.S. Pat. No. 4,991,217 incorporated by reference herein. These clients can take numerous forms (desktop PC, laptop PC, PDA, etc.) having varying speech signal processing and communications capability. Thus, from the server side perspective, it is not easy to assure uniform treatment of all users accessing a voice-enabled web page, since such users may have significantly disparate word recognition and error rate performances. While a prior art reference to Gould et al.—U.S. Pat. No. 5,915,236—discusses generally the notion of tailoring a recognition process to a set of available computational resources, it does not address or attempt to solve the issue of how to optimize resources in a distributed environment such as a client-server model. Again, to enable such voice-based technologies on a wide-spread scale it is far more preferable to have a system that harmonizes and accounts for discrepancies in individual systems so that even the thinnest client is supportable, and so that all users are able to interact in a satisfactory manner with the remote server running the e-commerce, e-support and/or remote learning application.

Two references that refer to a distributed approach for speech recognition include U.S. Pat. Nos. 5,956,683 and 5,960,399 incorporated by reference herein. In the first of these, U.S. Pat. 5,956,683—Distributed Voice Recognition System. (assigned to Qualcomm) an implementation of a distributed voice recognition system between a telephony-based handset and a remote station is described. In this implementation, all of the word recognition operations seem to take place at the handset. This is done since the patent describes the benefits that result from locating of the system for acoustic feature extraction at the portable or cellular phone in order to limit degradation of the acoustic features due to quantization distortion resulting from the narrow bandwidth telephony channel. This reference therefore does not address the issue of how to ensure adequate performance for a very thin client platform. Moreover, it is difficult to determine, how, if at all, the system can perform real-time word recognition, and there is no meaningful description of how to integrate the system with a natural language processor.

The second of these references—U.S. Pat No. 5,960,399—Client/Server Speech Processor/Recognizer (assigned to GTE) describes the implementation of a HMM-based distributed speech recognition system. This reference is not instructive in many respects, however, including how to optimize acoustic feature extraction for a variety of client platforms, such as by performing a partial word recognition process where appropriate. Most importantly, there is only a description of a primitive server-based recognizer that only recognizes the user's speech and simply returns certain keywords such as the user's name and travel destination to fill out a dedicated form on the user's machine. Also, the streaming of the acoustic parameters does not appear to be implemented in real-time as it can only take place after silence is detected. Finally, while the reference mentions the possible use of natural language processing (column **9**) there is no explanation of how such function might be implemented in a real-time fashion to provide an interactive feel for the user.

SUMMARY OF THE INVENTION

An object of the present invention, therefore, is to provide an improved system and method for overcoming the limitations of the prior art noted above;

A primary object of the present invention is to provide a word and phrase recognition system that is flexibly and optimally distributed across a client/platform computing architecture, so that improved accuracy, speed and uniformity can be achieved for a wide group of users;

A further object of the present invention is to provide a speech recognition system that efficiently integrates a distributed word recognition system with a natural language processing system, so that both individual words and entire speech utterances can be quickly and accurately recognized in any number of possible languages;

A related object of the present invention is to provide an efficient query response system so that an extremely accurate, real-time set of appropriate answers can be given in response to speech-based queries;

Yet another object of the present invention is to provide an interactive, real-time instructional/learning system that is distributed across a client/server architecture, and permits a real-time question/answer session with an interactive character;

A related object of the present invention is to implement such interactive character with an articulated response capability so that the user experiences a human-like interaction;

Still a further object of the present invention is to provide an INTERNET website with speech processing capability so that voice based data and commands can be used to interact with such site, thus enabling voice-based e-commerce and e-support services to be easily scaleable;

Another object is to implement a distributed speech recognition system that utilizes environmental variables as part of the recognition process to improve accuracy and speed;

A further object is to provide a scaleable query/response database system, to support any number of query topics and users as needed for a particular application and instantaneous demand;

Yet another object of the present invention is to provide a query recognition system that employs a two-step approach, including a relatively rapid first step to narrow down the list of potential responses to a smaller candidate set, and a second more computationally intensive second step to identify the best choice to be returned in response to the query from the candidate set;

A further object of the present invention is to provide a natural language processing system that facilitates query recognition by extracting lexical components of speech utterances, which components can be used for rapidly identifying a candidate set of potential responses appropriate for such speech utterances;

Another related object of the present invention is to provide a natural language processing system that facilitates query recognition by comparing lexical components of speech utterances with a candidate set of potential response to provide an extremely accurate best response to such query.

One general aspect of the present invention, therefore, relates to a natural language query system (NLQS) that offers a fully interactive method for answering user's questions over a distributed network such as the INTERNET or a local intranet. This interactive system when implemented over the worldwide web (WWW) services of the INTERNET functions so that a client or user can ask a question in a natural language such as English, French, German or Spanish and receive the appropriate answer at his or her personal computer also in his or her native natural language.

The system is distributed and consists of a set of integrated software modules at the client's machine and another set of integrated software programs resident on a server or set of servers. The client-side software program is comprised of a speech recognition program, an agent and its control program, and a communication program. The server-side program is comprised of a communication program, a natural language engine (NLE), a database processor (DBProcess), an interface program for interfacing the DBProcess with the NLE, and a SQL database. In addition, the client's machine is equipped with a microphone and a speaker. Processing of the speech utterance is divided between the client and server side so as to optimize processing and transmission latencies, and so as to provide support for even very thin client platforms.

In the context of an interactive learning application, the system is specifically used to provide a single-best answer to a user's question. The question that is asked at the client's machine is articulated by the speaker and captured by a microphone that is built in as in the case of a notebook computer or is supplied as a standard peripheral attachment. Once the question is captured, the question is processed partially by NLQS client-side software resident in the client's machine. The output of this partial processing is a set of speech vectors that are transported to the server via the INTERNET to complete the recognition of the user's questions. This recognized speech is then converted to text at the server.

After the user's question is decoded by the speech recognition engine (SRE) located at the server, the question is converted to a structured query language (SQL) query. This query is then simultaneously presented to a software process within the server called DBProcess for preliminary processing and to a Natural Language Engine (NLE) module for extracting the noun phrases (NP) of the user's question. During the process of extracting the noun phrase within the NLE, the tokens of the users' question are tagged. The tagged tokens are then grouped so that the NP list can be determined. This information is stored and sent to the DBProcess process.

In the DBProcess, the SQL query is fully customized using the NP extracted from the user's question and other environment variables that are relevant to the application. For example, in a training application, the user's selection of course, chapter and or section would constitute the environment variables. The SQL query is constructed using the extended SQL Full-Text predicates—CONTAINS, FREETEXT, NEAR, AND. The SQL query is next sent to the Full-Text search engine within the SQL database, where a Full-Text search procedure is initiated. The result of this search procedure is recordset of answers. This recordset contains stored questions that are similar linguistically to the user's question. Each of these stored questions has a paired answer stored in a separate text file, whose path is stored in a table of the database.

The entire recordset of returned stored answers is then returned to the NLE engine in the form of an array. Each stored question of the array is then linguistically processed sequentially one by one. This linguistic processing constitutes the second step of a 2-step algorithm to determine the single best answer to the user's question. This second step proceeds as follows: for each stored question that is returned in the recordset, a NP of the stored question is compared with the NP of the user's question. After all stored questions of the array are compared with the user's question, the stored question that yields the maximum match with the user's question is selected as the best possible stored question that matches the user's question. The metric that is used to determine the best possible stored question is the number of noun phrases.

The stored answer that is paired to the best-stored question is selected as the one that answers the user's question. The ID tag of the question is then passed to the DBProcess. This DBProcess returns the answer which is stored in a file.

A communication link is again established to send the answer back to the client in compressed form. The answer once received by the client is decompressed and articulated to the user by the text-to-speech engine. Thus, the invention can be used in any number of different applications involving interactive learning systems, INTERNET related commerce sites, INTERNET search engines, etc.

Computer-assisted instruction environments often require the assistance of mentors or live teachers to answer questions from students. This assistance often takes the form of organizing a separate pre-arranged forum or meeting time that is set aside for chat sessions or live call-in sessions so that at a scheduled time answers to questions may be provided. Because of the time immediacy and the on-demand or asynchronous nature of on-line training where a student may log on and take instruction at any time and at any location, it is important that answers to questions be provided in a timely and cost-effective manner so that the user or student can derive the maximum benefit from the material presented.

This invention addresses the above issues. It provides the user or student with answers to questions that are normally channeled to a live teacher or mentor. This invention provides a single-best answer to questions asked by the student. The student asks the question in his or her own voice in the language of choice. The speech is recognized and the answer to the question is found using a number of technologies including distributed speech recognition, full-text search database processing, natural language processing and text-to-speech technologies. The answer is presented to the user, as in the case of a live teacher, in an articulated manner by an agent that mimics the mentor or teacher, and in the language of choice—English, French, German, Japanese or other natural spoken language. The user can choose the agent's gender as well as several speech parameters such as pitch, volume and speed of the character's voice.

Other applications that benefit from NLQS are e-commerce applications. In this application, the user's query for a price of a book, compact disk or for the availability of any item that is to be purchased can be retrieved without the need to pick through various lists on successive web pages. Instead, the answer is provided directly to the user without any additional user input.

Similarly, it is envisioned that this system can be used to provide answers to frequently-asked questions (FAQs), and as a diagnostic service tool for e-support. These questions are typical of a give web site and are provided to help the user find information related to a payment procedure or the specifications of, or problems experienced with a product/service. In all of these applications, the NLQS architecture can be applied.

A number of inventive methods associated with these architectures are also beneficially used in a variety of INTERNET related applications.

Although the inventions are described below in a set of preferred embodiments, it will be apparent to those skilled in the art the present inventions could be beneficially used in many environments where it is necessary to implement fast, accurate speech recognition, and/or to provide a human-like dialog capability to an intelligent system.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram of a preferred embodiment of a natural language query system (NLQS) of the present invention, which is distributed across a client/server computing architecture, and can be used as an interactive learning system, an e-commerce system, an e-support system, and the like;

FIG. **2** is a block diagram of a preferred embodiment of a client side system, including speech capturing modules, partial speech processing modules, encoding modules, transmission modules, agent control modules, and answer/voice feedback modules that can be used in the aforementioned NLQS;

FIG. **2**—**2** is a block diagram of a preferred embodiment of a set of initialization routines and procedures used for the client side system of FIG. **2**;

FIG. **3** is a block diagram of a preferred embodiment of a set of routines and procedures used for handling an iterated set of speech utterances on the client side system of FIG. **2**, transmitting speech data for such utterances to a remote server, and receiving appropriate responses back from such server;

FIG. **4** is a block diagram of a preferred embodiment of a set of initialization routines and procedures used for un-initializing the client side system of FIG. **2**;

FIG. **4**A is a block diagram of a preferred embodiment of a set of routines and procedures used for implementing a distributed component of a speech recognition module for the server side system of FIG. **5**;

FIG. **4**B is a block diagram of a preferred set of routines and procedures used for implementing an SQL query builder for the server side system of FIG. **5**;

FIG. **4**C is a block diagram of a preferred embodiment of a set of routines and procedures used for implementing a database control process module for the server side system of FIG. **5**;

FIG. **4**D is a block diagram of a preferred embodiment of a set of routines and procedures used for implementing a natural language engine that provides query formulation support, a query response module, and an interface to the database control process module for the server side system of FIG. **5**;

FIG. **5** is a block diagram of a preferred embodiment of a server side system, including a speech recognition module to complete processing of the speech utterances, environmental and grammar control modules, query formulation modules, a natural language engine, a database control module, and a query response module that can be used in the aforementioned NLQS;

FIG. **6** illustrates the organization of a full-text database used as part of server side system shown in FIG. **5**;

FIG. **7**A illustrates the organization of a full-text database course table used as part of server side system shown in FIG. **5** for an interactive learning embodiment of the present invention;

FIG. **7**B illustrates the organization of a full-text database chapter table used as part of server side system shown in FIG. **5** for an interactive learning embodiment of the present invention;

FIG. **7**C describes the fields used in a chapter table used as part of server side system shown in FIG. **5** for an interactive learning embodiment of the present invention;

FIG. **7**D describes the fields used in a section table used as part of server side system shown in FIG. **5** for an interactive learning embodiment of the present invention;

FIG. **8** is a flow diagram of a first set of operations performed by a preferred embodiment of a natural language engine on a speech utterance including Tokenization, Tagging and Grouping;

FIG. **9** is a flow diagram of the operations performed by a preferred embodiment of a natural language engine on a speech utterance including stemming and Lexical Analysis

FIG. **10** is a block diagram of a preferred embodiment of a SQL database search and support system for the present invention;

FIGS. **11**A–**11**C are flow diagrams illustrating steps performed in a preferred two step process implemented for query recognition by the NLQS of FIG. **2**;

FIG. **12** is an illustration of another embodiment of the present invention implemented as part of a Web-based speech based learning/training System;

FIGS. **13**–**17** are illustrations of another embodiment of the present invention implemented as part of a Web-based e-commerce system;

FIG. **18** is an illustration of another embodiment of the present invention implemented as part of a voice-based Help Page for an E-Commerce Web Site.

DETAILED DESCRIPTION OF THE INVENTION

Overview

As alluded to above, the present inventions allow a user to ask a question in a natural language such as English, French, German, Spanish or Japanese at a client computing system (which can be as simple as a personal digital assistant or cell-phone, or as sophisticated as a high end desktop PC) and receive an appropriate answer from a remote server also in his or her native natural language. As such, the embodiment of the invention shown in FIG. **1** is beneficially used in what can be generally described as a Natural Language Query System (NLQS) **100**, which is configured to interact on a real-time basis to give a human-like dialog capability/experience for e-commerce, e-support, and e-learning applications.

The processing for NLQS **100** is generally distributed across a client side system **150**, a data link **160**, and a server-side system **180**. These components are well known in the art, and in a preferred embodiment include a personal computer system **150**, an INTERNET connection **160**A, **160**B, and a larger scale computing system **180**. It will be understood by those skilled in the art that these are merely exemplary components, and that the present invention is by no means limited to any particular implementation or combination of such systems. For example, client-side system **150** could also be implemented as a computer peripheral, a PDA, as part of a cell-phone, as part of an INTERNET-adapted appliance, an INTERNET linked kiosk, etc. Similarly, while an INTERNET connection is depicted for data link **160**A, it is apparent that any channel that is suitable for carrying data between client system **150** and server system **180** will suffice, including a wireless link, an RF link, an IR link, a LAN, and the like. Finally, it will be further appreciated that server system **180** may be a single, large-scale system, or a collection of smaller systems interlinked to support a number of potential network users.

Initially speech input is provided in the form of a question or query articulated by the speaker at the client's machine or personal accessory as a speech utterance. This speech utterance is captured and partially processed by NLQS client-

side software **155** resident in the client's machine. To facilitate and enhance the human-like aspects of the interaction, the question is presented in the presence of an animated character **157** visible to the user who assists the user as a personal information retriever/agent. The agent can also interact with the user using both visible text output on a monitor/display (not shown) and/or in audible form using a text to speech engine **159**. The output of the partial processing done by SRE **155** is a set of speech vectors that are transmitted over communication channel **160** that links the user's machine or personal accessory to a server or servers via the INTERNET or a wireless gateway that is linked to the INTERNET as explained above. At server **180**, the partially processed speech signal data is handled by a server-side SRE **182**, which then outputs recognized speech text corresponding to the user's question. Based on this user question related text, a text-to-query converter **184** formulates a suitable query that is used as input to a database processor **186**. Based on the query, database processor **186** then locates and retrieves an appropriate answer using a customized SQL query from database **188**. A Natural Language Engine **190** facilitates structuring the query to database **188**. After a matching answer to the user's question is found, the former is transmitted in text form across data link **160**B, where it is converted into speech by text to speech engine **159**, and thus expressed as oral feedback by animated character agent **157**.

Because the speech processing is broken up in this fashion, it is possible to achieve real-time, interactive, human-like dialog consisting of a large, controllable set of questions/answers. The assistance of the animated agent **157** further enhances the experience, making it more natural and comfortable for even novice users. To make the speech recognition process more reliable, context-specific grammars and dictionaries are used, as well as natural language processing routines at NLE **190**, to analyze user questions lexically. While context-specific processing of speech data is known in the art (see e.g., U.S. Pat Nos. 5,960,394, 5,867,817, 5,758,322 and 5,384,892 incorporated by reference herein) the present inventors are unaware of any such implementation as embodied in the present inventions. The text of the user's question is compared against text of other questions to identify the question posed by the user by DB processor/engine (DBE) **186**. By optimizing the interaction and relationship of the SR engines **155** and **182**, the NLP routines **190**, and the dictionaries and grammars, an extremely fast and accurate match can be made, so that a unique and responsive answer can be provided to the user.

On the server side **180**, interleaved processing further accelerates the speech recognition process. In simplified terms, the query is presented simultaneously both to NLE **190** after the query is formulated, as well as to DBE **186**. NLE **190** and SRE **182** perform complementary functions in the overall recognition process. In general, SRE **182** is primarily responsible for determining the identity of the words articulated by the user, while NLE **190** is responsible for the linguistic morphological analysis of both the user's query and the search results returned after the database query.

After the user's query is analyzed by NLE **190** some parameters are extracted and sent to the DBProcess. Additional statistics are stored in an array for the 2^{nd} step of processing. During the 2^{nd} step of 2-step algorithm, the recordset of preliminary search results are sent to the NLE **160** for processing. At the end of this 2^{nd} step, the single question that matches the user's query is sent to the DBProcess where further processing yields the paired answer that is paired with the single best stored question.

Thus, the present invention uses a form of natural language processing (NLP) to achieve optimal performance in a speech based web application system. While NLP is known in the art, prior efforts in Natural Language Processing (NLP) work nonetheless have not been well integrated with Speech Recognition (SR) technologies to achieve reasonable results in a web-based application environment. In speech recognition, the result is typically a lattice of possible recognized words each with some probability of fit with the speech recognizer. As described before, the input to a typical NLP system is typically a large linguistic unit. The NLP system is then charged with the parsing, understanding and indexing of this large linguistic unit or set of transcribed utterances. The result of this NLP process is to understand lexically or morphologically the entire linguistic unit as opposed to word recognition. Put another way, the linguistic unit or sentence of connected words output by the SRE has to be understood lexically, as opposed to just being "recognized".

As indicated earlier, although speech recognition technology has been available for several years, the technical requirements for the NLQS invention have placed severe restrictions on the specifications for the speech recognition accuracy that is required for an application that combines speech recognition and natural language processing to work satisfactorily. In realizing that even with the best of conditions, it might be not be possible to achieve the perfect 100% speech recognition accuracy that is required, the present invention employs an algorithm that balances the potential risk of the speech recognition process with the requirements of the natural language processing so that even in cases where perfect speech recognition accuracy is not achieved for each word in the query, the entire query itself is nonetheless recognized with sufficient accuracy.

This recognition accuracy is achieved even while meeting very stringent user constraints, such as short latency periods of 3 to 5 seconds (ideally—ignoring transmission latencies which can vary) for responding to a speech-based query, and for a potential set of 100–250 query questions. This quick response time gives the overall appearance and experience of a real-time discourse that is more natural and pleasant from the user's perspective. Of course, non-real time applications, such as translation services for example, can also benefit from the present teachings as well, since a centralized set of HMMs, grammars, dictionaries, etc., are maintained.

General Aspects of Speech Recognition Used in the Present Inventions

General background information on speech recognition can be found in the prior art references discussed above and incorporated by reference herein. Nonetheless, a discussion of some particular exemplary forms of speech recognition structures and techniques that are well-suited for NLQS **100** is provided next to better illustrate some of the characteristics, qualities and features of the present inventions.

Speech recognition technology is typically of two types—speaker independent and speaker dependent. In speaker-dependent speech recognition technology, each user has a voice file in which a sample of each potentially recognized word is stored. Speaker-dependent speech recognition systems typically have large vocabularies and dictionaries making them suitable for applications as dictation and text transcribing. It follows also that the memory and processor resource requirements for the speaker-dependent can be and are typically large and intensive.

Conversely speaker-independent speech recognition technology allows a large group of users to use a single vocabulary file. It follows then that the degree of accuracy that can be achieved is a function of the size and complexity of the grammars and dictionaries that can be supported for a given language. Given the context of applications for which NLQS, the use of small grammars and dictionaries allow speaker independent speech recognition technology to be implemented in NLQS.

The key issues or requirements for either type—speaker-independent or speaker-dependent, are accuracy and speed. As the size of the user dictionaries increase, the speech recognition accuracy metric—word error rate (WER) and the speed of recognition decreases. This is so because the search time increases and the pronunciation match becomes more complex as the size of the dictionary increases.

The basis of the NLQS speech recognition system is a series of Hidden Markov Models (HMN), which, as alluded to earlier, are mathematical models used to characterize any time varying signal. Because parts of speech are considered to be based on an underlying sequence of one or more symbols, the HMM models corresponding to each symbol are trained on vectors from the speech waveforms. The Hidden Markov Model is a finite set of states, each of which is associated with a (generally multi-dimensional) probability distribution. Transitions among the states are governed by a set of probabilities called transition probabilities. In a particular state an outcome or observation can be generated, according to an associated probability distribution. This finite state machine changes state once every time unit, and each time t such that a state j is entered, a spectral parameter vector O_t is generated with probability density $B_j(O_t)$. It is only the outcome, not the state which is visible to an external observer and therefore states are "hidden" to the outside; hence the name Hidden Markov Model.

In isolated speech recognition, it is assumed that the sequence of observed speech vectors corresponding to each word can each be described by a Markov model as follows:

$$O=o_1, o_2, \ldots o_t \quad (1\text{-}1)$$

where o_t is a speech vector observed at time t. The isolated word recognition then is to compute:

$$\arg\max \{P(w_i|O)\} \quad (1\text{-}2)$$

By using Bayes' Rule,

$$\{P(w_i|O)\}=[P(O|w_i)P(w_i)]/P(O) \quad (1\text{-}3)$$

In the general case, the Markov model when applied to speech also assumes a finite state machine which changes state once every time unit and each time that a state j is entered, a speech vector o_t is generated from the probability density $b_j(o_t)$. Furthermore, the transition from state i to state j is also probabilistic and is governed by the discrete probability a_{ij}.

For a state sequence X, the joint probability that O is generated by the model M moving through a state sequence X is the product of the transition probabilities and the output probabilities. Only the observation sequence is known—the state sequence is hidden as mentioned before.

Given that X is unknown, the required likelihood is computed by summing over all possible state sequences X=x(1), x(2), x(3), . . . x(t), that is

$$P(O|M)=\Sigma\{a_{x(0)x(1)}\Pi b(x)(o_t)a_{x(t)x(t+1)}\}$$

Given a set of models M_i, corresponding to words w_i equation 1-2 is solved by using 1-3 and also by assuming that:

$$P(O|w_i)=P(O|M_i)$$

All of this assumes that the parameters $\{a_{ij}\}$ and $\{b_j(o_t)\}$ are known for each model M_i. This can be done, as explained earlier, by using a set of training examples corresponding to a particular model. Thereafter, the parameters of that model can be determined automatically by a robust and efficient re-estimation procedure. So if a sufficient number of representative examples of each word are collected, then a HMM can be constructed which simply models all of the many sources of variability inherent in real speech. This training is well-known in the art, so it is not described at length herein, except to note that the distributed architecture of the present invention enhances the quality of HMMs, since they are derived and constituted at the server side, rather than the client side. In this way, appropriate samples from users of different geographical areas can be easily compiled and analyzed to optimize the possible variations expected to be seen across a particular language to be recognized. Uniformity of the speech recognition process is also well-maintained, and error diagnostics are simplified, since each prospective user is using the same set of HMMs during the recognition process.

To determine the parameters of a HMM from a set of training samples, the first step typically is to make a rough guess as to what they might be. Then a refinement is done using the Baum-Welch estimation formulae. By these formulae, the maximum likelihood estimates of μ_j (where μ_j is mean vector and Σ_i is covariance matrix) is:

$$=\mu_j=\Sigma^T_{t-1}L_j(t)o_t/[\Sigma^T_{t-1}L_j(t)o_t]$$

A forward-backward algorithm is next used to calculate the probability of state occupation $L_j(t)$. If the forward probability α_i (t) for some model M with N states is defined as:

$$\alpha_j(t)=P(o_1, \ldots, o_t, x(t)=j|M)$$

This probability can be calculated using the recursion:

$$\alpha_j(t)=[\Sigma^{N-1}_{i-2}\alpha(t-1)a_{ij}]b_j(o_t)$$

Similarly the backward probability can be computed using the recursion:

$$\beta_j(t)=\Sigma^{N-1}_{j-2}a_{ij}b_j(o_{t+1})(t+1)$$

Realizing that the forward probability is a joint probability and the backward probability is a conditional probability, the probability of state occupation is the product of the two probabilities:

$$\alpha_j(t)\beta_j(t)=P(O, x(t)=j|M)$$

Hence the probability of being in state j at a time t is:

$$L_j(t)=1/P[\alpha_j(t)\beta_j(t)]$$

where P=P(O|M)

To generalize the above for continuous speech recognition, we assume the maximum likelihood state sequence where the summation is replaced by a maximum operation. Thus for a given model M, let ϕj (t) represent the maximum likelihood of observing speech vectors o_1 to o_t and being used in state j at time t:

$$\phi_i(t)=\max\{\phi j\ (t)(t-1)\alpha_{ij}\}\cdot\beta_j(o_t)$$

Expressing this logarithmically to avoid underflow, this likelihood becomes:

$$\psi_j(t)=\max\{\psi_i(t-1)+\log(\alpha_{ij})\}+\log(b_j(o_t))$$

This is also known as the Viterbi algorithm. It can be visualized as finding the best path through a matrix where the vertical dimension represents the states of the HMM and horizontal dimension represents frames of speech i.e. time. To complete the extension to connected speech recognition, it is further assumed that each HMM representing the underlying sequence is connected. Thus the training data for continuous speech recognition should consist of connected utterances; however, the boundaries between words do not have to be known.

To improve computational speed/efficiency, the Viterbi algorithm is sometimes extended to achieve convergence by using what is known as a Token Passing Model. The token passing model represents a partial match between the observation sequence o_1 to o_t and a particular model, subject to the constraint that the model is in state j at time t. This token passing model can be extended easily to connected speech environments as well if we allow the sequence of HMMs to be defined as a finite state network. A composite network that includes both phoneme-based HMMs and complete words can be constructed so that a single-best word can be recognized to form connected speech using word N-best extraction from the lattice of possibilities. This composite form of HMM-based connected speech recognizer is the basis of the NLQS speech recognizer module. Nonetheless, the present invention is not limited as such to such specific forms of speech recognizers, and can employ other techniques for speech recognition if they are otherwise compatible with the present architecture and meet necessary performance criteria for accuracy and speed to provide a real-time dialog experience for users.

The representation of speech for the present invention's HMM-based speech recognition system assumes that speech is essentially either a quasi-periodic pulse train (for voiced speech sounds) or a random noise source (for unvoiced sounds). It may be modeled as two sources—one a impulse train generator with pitch period P and a random noise generator which is controlled by a voice/unvoiced switch. The output of the switch is then fed into a gain function estimated from the speech signal and scaled to feed a digital filter H(z) controlled by the vocal tract parameter characteristics of the speech being produced. All of the parameters for this model—the voiced/unvoiced switching, the pitch period for voiced sounds, the gain parameter for the speech signal and the coefficient of the digital filter, vary slowly with time. In extracting the acoustic parameters from the user's speech input so that it can evaluated in light of a set of HMMs, cepstral analysis is typically used to separate the vocal tract information from the excitation information. The cepstrum of a signal is computed by taking the Fourier (or similar) transform of the log spectrum. The principal advantage of extracting cepstral coefficients is that they are de-correlated and the diagonal covariances can be used with HMMs. Since the human ear resolves frequencies non-linearly across the audio spectrum, it has been shown that a front-end that operates in a similar non-linear way improves speech recognition performance.

Accordingly, instead of a typical linear prediction-based analysis, the front-end of the NLQS speech recognition engine implements a simple, fast Fourier transform based filter bank designed to give approximately equal resolution on the Mel-scale. To implement this filter bank, a window of speech data (for a particular time frame) is transformed using a software based Fourier transform and the magnitude taken. Each FFT magnitude is then multiplied by the corresponding filter gain and the results accumulated. The cepstral coefficients that are derived from this filter-bank analysis at the front end are calculated during a first partial processing phase of the speech signal by using a Discrete Cosine Transform of the log filter bank amplitudes. These cepstral coefficients are called Mel-Frequency Cepstral Coefficients (MFCC) and they represent some of the speech parameters transferred from the client side to characterize the acoustic features of the user's speech signal. These parameters are chosen for a number of reasons, including the fact that they can be quickly and consistently derived even across systems of disparate capabilities (i.e., for everything from a low power PDA to a high powered desktop system), they give good discrimination, they lend themselves to a number of useful recognition related manipulations, and they are relatively small and compact in size so that they can be transported rapidly across even a relatively narrow band link. Thus, these parameters represent the least amount of information that can be used by a subsequent server side system to adequately and quickly complete the recognition process.

To augment the speech parameters an energy term in the form of the logarithm of the signal energy is added. Accordingly, RMS energy is added to the 12 MFCC's to make 13 coefficients. These coefficients together make up the partially processed speech data transmitted in compressed form from the user's client system to the remote server side.

The performance of the present speech recognition system is enhanced significantly by computing and adding time derivatives to the basic static MFCC parameters at the server side. These two other sets of coefficients—the delta and acceleration coefficients representing change in each of the 13 values from frame to frame (actually measured across several frames), are computed during a second partial speech signal processing phase to complete the initial processing of the speech signal, and are added to the original set of coefficients after the latter are received. These MFCCs together with the delta and acceleration coefficients constitute the observation vector O_t mentioned above that is used for determining the appropriate HMM for the speech data.

The delta and acceleration coefficients are computed using the following regression formula:

$$d_t=\Sigma^{\theta}_{\theta=1}[c_{t+\theta}-c_{t-\theta}]/2\Sigma^{\theta}_{\theta=1}\theta^2$$

where d_t is a delta coefficient at time t computed in terms of the corresponding static coefficients:

$$d_t=[c_{t+\theta}-c_{t-\theta}]/2\theta$$

In a typical stand-alone implementation of a speech recognition system, the entire SR engine runs on a single client. In other words, both the first and second partial processing phases above are executed by the same DSP (or microprocessor) running a ROM or software code routine at the client's computing machine.

In contrast, because of several considerations, specifically—cost, technical performance, and client hardware uniformity, the present NLQS system uses a partitioned or distributed approach. While some processing occurs on the client side, the main speech recognition engine runs on a centrally located server or number of servers. More specifically, as noted earlier, capture of the speech signals, MFCC vector extraction and compression are implemented on the client's machine during a first partial processing phase. The routine is thus streamlined and simple enough to be implemented within a browser program (as a plug in module, or a downloadable applet for example) for maxi-

mum ease of use and utility. Accordingly, even very "thin" client platforms can be supported, which enables the use of the present system across a greater number of potential sites. The primary MFCCs are then transmitted to the server over the channel, which, for example, can include a dial-up INTERNET connection, a LAN connection, a wireless connection and the like. After decompression, the delta and acceleration coefficients are computed at the server to complete the initial speech processing phase, and the resulting observation vectors O_t are also determined.

General Aspects of Speech Recognition Engine

The speech recognition engine is also located on the server, and is based on a HTK based recognition network compiled from a word-level network, a dictionary and a set of HMMs. The recognition network consists of a set of nodes connected by arcs. Each node is either a HMM model instance or a word end. Each model node is itself a network consisting of states connected by arcs. Thus when fully compiled, a speech recognition network consists of HMM states connected by transitions. For an unknown input utterance with T frames, every path from the start node to the exit node of the network passes through T HMM states. Each of these paths has log probability which is computed by summing the log probability of each individual transition in the path and the log probability of each emitting state generating the corresponding observation. The function of the Viterbi decoder is find those paths through the network which have the highest log probability. This is found using the Token Passing algorithm. In a network that has many nodes, the computation time is reduced by only allowing propagation of those tokens which will have some chance of becoming winners. This process is called pruning.

Natural Language Processor

In a typical natural language interface to a database, the user enters a question in his/her natural language, for example, English. The system parses it and translates it to a query language expression. The system then uses the query language expression to process the query and if the search is successful, a recordset representing the results is displayed in English either formatted as raw text or in a graphical form. For a natural language interface to work well involves a number of technical requirements.

For example, it needs to be robust—in the sentence 'What's the departments turnover' it needs to decide that the word whats=what's=what is. And it also has to determine that departments=department's. In addition to being robust, the natural language interface has to distinguish between the several possible forms of ambiguity that may exist in the natural language—lexical, structural, reference and ellipsis ambiguity. All of these requirements, in addition to the general ability to perform basic linguistic morphological operations of tokenization, tagging and grouping, are implemented within the present invention.

Tokenization is implemented by a text analyzer which treats the text as a series of tokens or useful meaningful units that are larger than individual characters, but smaller than phrases and sentences. These include words, separable parts of words, and punctuation. Each token is associated with an offset and a length. The first phase of tokenization is the process of segmentation which extracts the individual tokens from the input text and keeps track of the offset where each token originated in the input text. The tokenizer output lists the offset and category for each token. In the next phase of the text analysis, the tagger uses a built-in morphological analyzer to look up each word/token in a phrase or sentence and internally lists all parts of speech. The output is the input string with each token tagged with a parts of speech notation. Finally the grouper which functions as a phrase extractor or phrase analyzer, determines which groups of words form phrases. These three operations which are the foundations for any modern linguistic processing schemes, are fully implemented in optimized algorithms for determining the single-best possible answer to the user's question.

SQL Database and Full-Text Query

Another key component of present system is a SQL-database. This database is used to store text, specifically the answer-question pairs are stored in full-text tables of the database. Additionally, the full-text search capability of the database allows full-text searches to be carried out.

While a large portion of all digitally stored information is in the form of unstructured data, primarily text, it is now possible to store this textual data in traditional database systems in character-based columns such as varchar and text. In order to effectively retrieve textual data from the database, techniques have to be implemented to issue queries against textual data and to retrieve the answers in a meaningful way where it provides the answers as in the case of the NLQS system.

There are two major types of textual searches: Property—This search technology first applies filters to documents in order to extract properties such as author, subject, type, word count, printed page count, and time last written, and then issues searches against those properties; Full-text—this search technology first creates indexes of all non-noise words in the documents, and then uses these indexes to support linguistic searches and proximity searches.

Two additional technologies are also implemented in this particular RDBMs: SQL Server also have been integrated: A Search service—a full-text indexing and search service that is called both index engine and search, and a parser that accepts full-text SQL extensions and maps them into a form that can be processed by the search engine.

The four major aspects involved in implementing full-text retrieval of plain-text data from a full-text-capable database are: Managing the definition of the tables and columns that are registered for full-text searches; Indexing the data in registered columns—the indexing process scans the character streams, determines the word boundaries (this is called word breaking), removes all noise words (this also is called stop words), and then populates a full-text index with the remaining words; Issuing queries against registered columns for populated full-text indexes; Ensuring that subsequent changes to the data in registered columns gets propagated to the index engine to keep the full-text indexes synchronized.

The underlying design principle for the indexing, querying, and synchronizing processes is the presence of a full-text unique key column (or single-column primary key) on all tables registered for full-text searches. The full-text index contains an entry for the non-noise words in each row together with the value of the key column for each row.

When processing a full-text search, the search engine returns to the database the key values of the rows that match the search criteria.

The full-text administration process starts by designating a table and its columns of interest for full-text search. Customized NLQS stored procedures are used first to register tables and columns as eligible for full-text search. After that, a separate request by means of a stored procedure is issued to populate the full-text indexes. The result is that the

underlying index engine gets invoked and asynchronous index population begins. Full-text indexing tracks which significant words are used and where they are located. For example, a full-text index might indicate that the word "NLQS" is found at word number **423** and word number **982** in the Abstract column of the DevTools table for the row associated with a ProductID of 6. This index structure supports an efficient search for all items containing indexed words as well as advanced search operations, such as phrase searches and proximity searches. (An example of a phrase search is looking for "white elephant," where "white" is followed by "elephant". An example of a proximity search is looking for "big" and "house" where "big" occurs near "house".) To prevent the full-text index from becoming bloated, noise words such as "a," "and," and "the" are ignored.

Extensions to the Transact-SQL language are used to construct full-text queries. The two key predicates that are used in the NLQS are CONTAINS and FREETEXT.

The CONTAINS predicate is used to determine whether or not values in full-text registered columns contain certain words and phrases. Specifically, this predicate is used to search for:

A word or phrase.

The prefix of a word or phrase.

A word or phrase that is near another.

A word that is an inflectional form of another (for example, "drive" is the inflectional stem of "drives," "drove," "driving," and "driven").

A set of words or phrases, each of which is assigned a different weighting.

The relational engine within SQL Server recognizes the CONTAINS and FREETEXT predicates and performs some minimal syntax and semantic checking, such as ensuring that the column referenced in the predicate has been registered for full-text searches. During query execution, a full-text predicate and other relevant information are passed to the full-text search component. After further syntax and semantic validation, the search engine is invoked and returns the set of unique key values identifying those rows in the table that satisfy the full-text search condition. In addition to the FREETEXT and CONTAINS, other predicates such as AND, LIKE, NEAR are combined to create the customized NLQS SQL construct.

Full-Text Query Architecture of the SQL Database

The full-text query architecture is comprised of the following several components—Full-Text Query component, the SQL Server Relational Engine, the Full-Text provider and the Search Engine.

The Full-Text Query component of the SQL database accept a full-text predicate or rowset-valued function from the SQL Server; transform parts of the predicate into an internal format, and sends it to Search Service, which returns the matches in a rowset. The rowset is then sent back to SQL Server. SQL Server uses this information to create the resultset that is then returned to the submitter of the query.

The SQL Server Relational Engine accepts the CONTAINS and FREETEXT predicates as well as the CONTAINSTABLE( ) and FREETEXTTABLE( ) rowset-valued functions. During parse time, this code checks for conditions such as attempting to query a column that has not been registered for full-text search. If valid, then at run time, the ft_search_condition and context information is sent to the full-text provider. Eventually, the full-text provider returns a rowset to SQL Server, which is used in any joins (specified or implied) in the original query. The Full-Text Provider parses and validates the ft_search_condition, constructs the appropriate internal representation of the full-text search condition, and then passes it to the search engine. The result is returned to the relational engine by means of a rowset of rows that satisfy ft_search_condition.

Client Side System **150**

The architecture of client-side system **150** of Natural Language Query System **100** is illustrated in greater detail in FIG. **2**. Referring to FIG. **2**, the three main processes effectuated by Client System **150** are illustrated as follows: Initialization process **200**A consisting of SRE **201**, Communication **202** and Microsoft (NS) Agent **203** routines; an iterative process **200**B consisting of two sub-routines: a) Receive User Speech **208**—made up of SRE **204** and Communication **205**; and b) Receive Answer from Server **207**—made up of MS Speak Agent **206**, Communication **209**, Voice data file **210** and Text to Speech Engine **211**. Finally, un-initialization process **200**C is made up of three sub-routines: SRE **212**, Communication **213**, and MS Agent **214**. Each of the above three processes are described in detail in the following paragraphs. It will be appreciated by those skilled in the art that the particular implementation for such processes and routines will vary from client platform to platform, so that in some environments such processes may be embodied in hard-coded routines executed by a dedicated DSP, while in others they may be embodied as software routines executed by a shared host processor, and in still others a combination of the two may be used.

Initialization at Client System **150**

The initialization of the Client System **150** is illustrated in FIG. **2**—**2** and is comprised generally of 3 separate initializing processes: client-side Speech Recognition Engine **220**A, MS Agent **220**B and Communication processes **220**C.

Initialization of Speech Recognition Engine **220**A

Speech Recognition Engine **155** is initialized and configured using the routines shown in **220**A. First, an SRE COM Library is initialized. Next, memory **220** is allocated to hold Source and Coder objects, are created by a routine **221**. Loading of configuration file **221**A from configuration data file **221**B also takes place at the same time that the SRE Library is initialized. In configuration file **221**B, the type of the input of Coder and the type of the output of the Coder are declared. The structure, operation, etc. of such routines are well-known in the art, and they can be implemented using a number of fairly straightforward approaches. Accordingly, they are not discussed in detail herein. Next, Speech and Silence components of an utterance are calibrated using a routine **222**, in a procedure that is also well-known in the art. To calibrate the speech and silence components, the user preferably articulates a sentence that is displayed in a text box on the screen. The SRE library then estimates the noise and other parameters required to find e silence and speech elements of future user utterances.

Initialization of MS Agent **220**B

The software code used to initialize and set up a MS Agent **220**B is also illustrated in FIG. **2**—**2**. The MS Agent **220**B routine is responsible for coordinating and handling the actions of the animated agent **157** (FIG. **1**). This initialization thus consists of the following steps:

1. Initialize COM library **223**. This part of the code initializes the COM library, which is required to use ActiveX Controls, which controls are well-known in the art.
2. Create instance of Agent Server **224**—this part of the code creates an instance of Agent ActiveX control.
3. Loading of MS Agent **225**—this part of the code loads MS Agent character from a specified file **225**A containing general parameter data for the Agent Character, such as the overall appearance, shape, size, etc.
4. Get Character Interface **226**—this part of the code gets an appropriate interface for the specified character; for example, characters may have different control/interaction capabilities that can be presented to the user.
5. Add Commands to Agent Character Option **227**—this part of the code adds commands to an Agent Properties sheet, which sheet can be accessed by clicking on the icon that appears in the system tray, when the Agent character is loaded e.g., that the character can Speak, how he/she moves, TTS Properties, etc.
6. Show the Agent Character **228**—this part of the code displays the Agent character on the screen so it can be seen by the user;
7. AgentNotifySink—to handle events. This part of the code creates AgentNotifySink object **229**, registers it at **230** and then gets the Agent Properties interface **231**. The property sheet for the Agent character is assigned using routine **232**.
8. Do Character Animations **233**—This part of the code plays specified character animations to welcome the user to NLQS **100**.

The above then constitutes the entire sequence required to initialize the MS Agent. As with the SRE routines, the MS Agent routines can be implemented in any suitable and conventional fashion by those skilled in the art based on the present teachings. The particular structure, operation, etc. of such routines is not critical, and thus they are not discussed in detail herein.

In a preferred embodiment, the MS Agent is configured to have an appearance and capabilities that are appropriate for the particular application. For instance, in a remote learning application, the agent has the visual form and mannerisms/attitude/gestures of a college professor. Other visual props (blackboard, textbook, etc.) may be used by the agent and presented to the user to bring to mind the experience of being in an actual educational environment. The characteristics of the agent may be configured at the client side **150**, and/or as part of code executed by a browser program (not shown) in response to configuration data and commands from a particular web page. For example, a particular website offering medical services may prefer to use a visual image of a doctor. These and many other variations will be apparent to those skilled in the art for enhancing the human-like, real-time dialog experience for users.

Initialization of Communication Link **160**A

The initialization of Communication Link **160**A is shown with reference to process **220**C FIG. **2**—**2**. Referring to FIG. **2**—**2**, this initialization consists of the following code components: Open INTERNET Connection **234**—this part of the code opens an INTERNET Connection and sets the parameter for the connection. Then Set Callback Status routine **235** sets the callback status so as to inform the user of the status of connection. Finally Start New HTTP INTERNET Session **236** starts a new INTERNET session. The details of Communications Link **160** and the set up process **220**C are not critical, and will vary from platform to platform. Again, in some cases, users may use a low-speed dial-up connection, a dedicated high speed switched connection (T**1** for example), an always-on xDSL connection, a wireless connection, and the like.

Iterative Processing of Queries/Answers

As illustrated in FIG. **3**, once initialization is complete, an iterative query/answer process is launched when the user presses the Start Button to initiate a query. Referring to FIG. **3**, the iterative query/answer process consists of two main sub-processes implemented as routines on the client side system **150**: Receive User Speech **240** and Receive User Answer **243**. The Receive User Speech **240** routine receives speech from the user (or another audio input source), while the Receive User Answer **243** routine receives an answer to the user's question in the form of text from the server so that it can be converted to speech for the user by text-to-speech engine **159**. As used herein, the term "query" is referred to in the broadest sense to refer, to either a question, a command, or some form of input used as a control variable by the system. For example, a, query may consist of a question directed to a particular topic, such as "what is a network" in the context of a remote learning application. In an e-commerce application a query might consist of a command to "list all books by Mark Twain" for example. Similarly, while the answer in a remote learning application consists of text that is rendered into audible form by the text to speech engine **159**, it could also be returned as another form of multi-media information, such as a graphic image, a sound file, a video file, etc. depending on the requirements of the particular application. Again, given the present teachings concerning the necessary structure, operation, functions, performance, etc., of the client-side Receive User Speech **240** and Receiver User Answer **243** routines, one of ordinary skill in the art could implement such in a variety of ways.

Receive User Speech—As illustrated in FIG. **3**, the Receive User Speech routine **240** consists of a SRE **241** and a Communication **242** process, both implemented again as routines on the client side system **150** for receiving and partially processing the user's utterance. SRE routine **241** uses a coder **248** which is prepared so that a coder object receives speech data from a source object. Next the Start Source **249** routine is initiated. This part of the code initiates data retrieval using the source Object which will in turn be given to the Coder object. Next, MFCC vectors **250** are extracted from the Speech utterance continuously until silence is detected. As alluded to earlier, this represents the first phase of processing of the input speech signal, and in a preferred embodiment, it is intentionally restricted to merely computing the MFCC vectors for the reasons already expressed above. These vectors include the 12 cepstral coefficients and the RMS energy term, for a total of 13 separate numerical values for the partially processed speech signal.

In some environments, nonetheless, it is conceivable that the MFCC delta parameters and MFCC acceleration parameters can also be computed at client side system **150**, depending on the computation resources available, the transmission bandwidth in data link **160**A available to server side system **180**, the speed of a transceiver used for carrying data in the data link, etc. These parameters can be determined automatically by client side system upon initializing SRE **155** (using some type of calibration routine to measure resources), or by direct user control, so that the partitioning of signal processing responsibilities can be optimized on a

case-by-case basis. In some applications, too, server side system **180** may lack the appropriate resources or routines for completing the processing of the speech input signal. Therefore, for some applications, the allocation of signal processing responsibilities may be partitioned differently, to the point where in fact both phases of the speech signal processing may take place at client side system **150** so that the speech signal is completely—rather than partially—processed and transmitted for conversion into a query at server side system **180**.

Again in a preferred embodiment, to ensure reasonable accuracy and real-time performance from a query/response perspective, sufficient resources are made available in a client side system so that 100 frames per second of speech data can be partially processed and transmitted through link **160**A. Since the least amount of information that is necessary to complete the speech recognition process (only 13 coefficients) is sent, the system achieves a real-time performance that is believed to be highly optimized, because other latencies (i.e., client-side computational latencies, packet formation latencies, transmission latencies) are minimized. It will be apparent that the principles of the present invention can be extended to other SR applications where some other methodology is used for breaking down the speech input signal by an SRE (i.e., non-MFCC based). The only criteria is that the SR processing be similarly dividable into multiple phases, and with the responsibility for different phases being handled on opposite sides of link **160**A depending on overall system performance goals, requirements and the like. This functionality of the present invention can thus be achieved on a system-by-system basis, with an expected and typical amount of optimization being necessary for each particular implementation.

Thus, the present invention achieves a response rate performance that is tailored in accordance with the amount of information that is computed, coded and transmitted by the client side system **150**. So in applications where real-time performance is most critical, the least possible amount of extracted speech data is transmitted to reduce these latencies, and, in other applications, the amount of extracted speech data that is processed, coded and transmitted can be varied.

Communication—transmit communication module **242** is used to implement the transport of data from the client to the server over the data link **160**A, which in a preferred embodiment is the INTERNET. As explained above, the data consists of encoded MFCC vectors that will be used at then server-side of the Speech Recognition engine to complete the speech recognition decoding. The sequence of the communication is as follows:

OpenHTTPRequest **251**—this part of the code first converts MFCC vectors to a stream of bytes, and then processes the bytes so that it is compatible with a protocol known as HTTP. This protocol is well-known in the art, and it is apparent that for other data links another suitable protocol would be used.

1. Encode MFCC Byte Stream **251**—this part of the code encodes the MFCC vectors, so that they can be sent to the server via HTTP.

2. Send data **252**—this part of the code sends MFCC vectors to the server using the INTERNET connection and the HTTP protocol.

Wait for the Server Response **253**—this part of the code monitors the data link **160**A a response from server side system **180** arrives. In summary, the MFCC parameters are extracted or observed on-the-fly from the input speech signal. They are then encoded to a HTTP byte stream and sent in a streaming fashion to the server before the silence is detected—i.e. sent to server side system **180** before the utterance is complete. This aspect of the invention also facilitates a real-time behavior, since data can be transmitted and processed even while the user is still speaking.

Receive Answer from Server **243** is comprised of the following modules as shown in FIG. **3**.: MS Agent **244**, Text-to-Speech Engine **245** and receive communication modules **246**. All three modules interact to receive the answer from server side system **180**. As illustrated in FIG. **3**, the receive communication process consists of three separate processes implemented as a receive routine on client side system **150**: a Receive the Best Answer **258** receives the best answer over data link **160**B (the HTTP communication channel). The answer is de-compressed at **259** and then the answer is passed by code **260** to the MS Agent **244**, where it is received by code portion **254**. A routine **255** then articulates the answer using text-to-speech engine **257**. Of course, the text can also be displayed for additional feedback purposes on a monitor used with client side system **150**. The text to speech engine uses a natural language voice data file **256** associated with it that is appropriate for the particular language application (i.e., English, French, German, Japanese, etc.). As explained earlier when the answer is something more than text, it can be treated as desired to provide responsive information to the user, such as with a graphics image, a sound, a video clip, etc.

Uninitialization

The un-initialization routines and processes are illustrated in FIG. **4**. Three functional modules are used for un-initializing the primary components of the client side system **150**; these include SRE **270**, Communications **271** and MS Agent **272** un-initializing routines. To un-initialize SRE **220**A, memory that was allocated in the initialization phase is de-allocated by code **273** and objects created during such initialization phase are deleted by code **274**. Similarly, as illustrated in FIG. **4**, to un-initialize Communications module **220**C the INTERNET connection previously established with the server is closed by code portion **275** of the Communication Un-initialization routine **271**. Next the INTERNET session created at the time of initialization is also closed by routine **276**. For the un-initialization of the MS Agent **220**B, as illustrated in FIG. **4**, MS Agent Un-initialization routine **272** first releases the Commands Interface **227** using routine **277**. This releases the commands added to the property sheet during loading of the agent character by routine **225**. Next the Character Interface initialized by routine **226** is released by routine **278** and the Agent is unloaded at **279**. The Sink Object Interface is then also released **280** followed by the release of the Property Sheet Interface **281**. The Agent Notify Sink **282** then un-registers the Agent and finally the Agent Interface **283** is released which releases all the resources allocated during initialization steps identified in FIG. **2**—**2**.

It will be appreciated by those skilled in the art that the particular implementation for such un-initialization processes and routines in FIG. **4** will vary from client platform to client platform, as for the other routines discussed above. The structure, operation, etc. of such routines are well-known in the art, and they can be implemented using a number of fairly straightforward approaches without undue effort. Accordingly, they are not discussed in detail herein.

DESCRIPTION OF SERVER SIDE SYSTEM **180**

Introduction

A high level flow diagram of the set of preferred processes implemented on server side system **180** of Natural Language

Query System **100** is illustrated in FIGS. **11**A through FIG. **11**C. In a preferred embodiment, this process consists of a two step algorithm for completing the processing of the speech input signal, recognizing the meaning of the user's query, and retrieving an appropriate answer/response for such query.

The 1st step as illustrated in FIG. **11**A can be considered a high-speed first-cut pruning mechanism, and includes the following operations: after completing processing of the speech input signal, the user's query is recognized at step **1101**, so that the text of the query is simultaneously sent to Natural Language Engine **190** (FIG. **1**) at step **1107**, and to DB Engine **186** (also FIG. **1**) at step **1102**. By "recognized" in this context it is meant that the user's query is converted into a text string of distinct native language words through the HMM technique discussed earlier.

At NLE **190**, the text string undergoes morphological linguistic processing at step **1108**: the string is tokenized the tags are tagged and the tagged tokens are grouped Next the noun phrases (NP) of the string are stored at **1109**, and also copied and transferred for use by DB Engine **186** during a DB Process at step **1110**. As illustrated in FIG. **11**A, the string corresponding to the user's query which was sent to the DB Engine **186** at **1102**, is used together with the NP received from NLE **190** to construct an SQL Query at step **1103**. Next, the SQL query is executed at step **1104**, and a record set of potential questions corresponding to the user's query are received as a result of a full-text search at **1105**, which are then sent back to NLE **190** in the form of an array at step **1106**.

As can be seen from the above, this first step on the server side processing acts as an efficient and fast pruning mechanism so that the universe of potential "hits" corresponding to the user's actual query is narrowed down very quickly to a manageable set of likely candidates in a very short period of time.

Referring to FIG. **11**B, in contrast to the first step above, the 2nd step can be considered as the more precise selection portion of the recognition process. It begins with linguistic processing of each of the stored questions in the array returned by the full-text search process as possible candidates representing the user's query. Processing of these stored questions continues in NLE **190** as follows: each question in the array of questions corresponding to the record set returned by the SQL full-text search undergoes morphological linguistic processing at step **1111**: in this operation, a text string corresponding to the retrieved candidate question is tokenized, the tags are tagged and the tagged tokens are grouped. Next, noun phrases of the string are computed and stored at step **1112**. This process continues iteratively at point **1113**, and the sequence of steps at **1118**,**1111**, **1112**, **1113** are repeated so that an NP for each retrieved candidate question is computed and stored. Once an NP is computed for each of the retrieved candidate questions of the array, a comparison is made between each such retrieved candidate question and the user's query based on the magnitude of the NP value at step **1114**. This process is also iterative in that steps **1114**, **1115**, **1116**, **1119** are repeated so that the comparison of the NP for each retrieved candidate question with that of the NP of the user's query is completed. When there are no more stored questions in the array to be processed at step **1117**, the stored question that has the maximum NP relative to the user's query, is identified at **1117**A as the stored question which best matches the user's query.

Notably, it can be seen that the second step of the recognition process is much more computationally intensive than the first step above, because several text strings are tokenized, and a comparison is made of several NPs. This would not be practical, nonetheless, if it were not for the fact that the first step has already quickly and efficiently reduced the candidates to be evaluated to a significant degree. Thus, this more computationally intensive aspect of the present invention is extremely valuable, however because it yields extremely high accuracy in the overall query recognition process. In this regard, therefore, this second step of the query recognition helps to ensure the overall accuracy of the system, while the first step helps to maintain a satisfactory speed that provides a real-time feel for the user.

As illustrated in FIG. **11**C, the last part of the query/response process occurs by providing an appropriate matching answer/response to the user. Thus, an identity of a matching stored question is completed at step **1120**. Next a file path corresponding to an answer of the identified matching question is extracted at step **1121**. Processing continues so that the answer is extracted from the file path at **1122** and finally the answer is compressed and sent to client side system **150** at step **1123**.

The discussion above is intended to convey a general overview of the primary components, operations, functions and characteristics of those portions of NLQS system **100** that reside on server side system **180**. The discussion that follows describes in more detail the respective sub-systems.

Software Modules Used in Server Side System **180**

The key software modules used on server-side system **180** of the NLQS system are illustrated in FIG. **5**. These include generally the following components: a Communication module **500**—identified as CommunicationServer ISAPI **500**A (which is executed by SRE Server-side **182** —FIG. **1** and is explained in more detail below), and a database process DBProcess module **501** (executed by DB Engine **186**—FIG. **1**). Natural language engine module **500**C (executed by NLE **190**—FIG. **1**) and an interface **500**B between the NLE process module **500**C and the DBProcess module **500**B. As shown here, CommunicationServerISAPI **500**A includes a server-side speech recognition engine and appropriate communication interfaces required between client side system **150** and server side system **180**. As further illustrated in FIG. **5**, server-side logic of Natural Language Query System **100** also can be characterized as including two dynamic link library components: CommunicationServerISAPI **500** and DBProcess **501**. The CommunicationServerIASPI **500** is comprised of 3 sub-modules: Server-side Speech Recognition Engine module **500**A; Interface module **500**B between Natural Language Engine modules **500**C and DBProcess **501**; and the Natural Language Engine modules **500**C.

DB Process **501** is a module whose primary function is to connect to a SQL database and to execute an SQL query that is composed in response to the user's query. In addition, this module interfaces with logic that fetches the correct answer from a file path once this answer is passed to it from the Natural Language Engine module **500**C.

Speech Recognition Sub-System **182** on Server-Side System **180**

The server side speech recognition engine module **500**A is a set of distributed components that perform the necessary functions and operations of speech recognition engine **182** (FIG. **1**) at server-side **180**. These components can be implemented as software routines that are executed by server side **180** in conventional fashion. Referring to FIG.

4A, a more detailed break out of the operation of the speech recognition components **600** at the server-side can be seen as follows:

Within a portion **601** of the server side SRE module **500**A, the binary MFCC vector byte stream corresponding to the speech signal's acoustic features extracted at client side system **150** and sent over the communication channel **160** is received. The MFCC acoustic vectors are decoded from the encoded HTTP byte stream as follows: Since the MFCC vectors contain embedded NULL characters, they cannot be transferred in this form to server side system **180** as such using HTTP protocol. Thus the MFCC vectors are first encoded at client-side **150** before transmission in such a way that all the speech data is converted into a stream of bytes without embedded NULL characters in the data. At the very end of the byte stream a single NULL character is introduced to indicate the termination of the stream of bytes to be transferred to the server over the INTERNET **160**A using HTTP protocol.

As explained earlier, to conserve latency time between the client and server, a smaller number of bytes Oust the **13** MFCC coefficients) are sent from client side system **150** to server side system **180**. This is done automatically for each platform to ensure uniformity, or can be tailored by the particular application environment—i.e., such as where it is determined that it will take less time to compute the delta and acceleration coefficients at the server (**26** more calculations), than it would take to encode them at the client, transmit them, and then decode them from the HTTP stream. In general, since server side system **180** is usually better equipped to calculate the MFCC delta and acceleration parameters, this is a preferable choice. Furthermore, there is generally more control over server resources compared to the client's resources, which means that future upgrades, optimizations, etc., can be disseminated and shared by all to make overall system performance more reliable and predictable. So, the present invention can accommodate even the worst-case scenario where the client's machine may be quite thin and may just have enough resources to capture the speech input data and do minimal processing.

Dictionary Preparation & Grammar Files

Referring to FIG. **4**A, within code block **605**, various options selected by the user (or gleaned from the user's status within a particular application) are received. For instance, in the case of a preferred remote learning system, Course, Chapter and/or Section data items are communicated. In the case of other applications (such as e-commerce) other data options are communicated, such as the Product Class, Product Category, Product Brand, etc. loaded for viewing within his/her browser. These selected options are based on the context experienced by the user during an interactive process, and thus help to limit and define the scope—i.e. grammars and dictionaries that will be dynamically loaded to speech recognition engine **182** (FIG. **1**) for Viterbi decoding during processing of the user speech utterance. For speech recognition to be optimized both grammar and dictionary files are used in a preferred embodiment. A Grammar file supplies the universe of available user queries; i.e., all the possible words that are to be recognized. The Dictionary file provides phonemes (the information of how a word is pronounced—this depends on the specific native language files that are installed—for example, UK English or US English) of each word contained in the grammar file. It is apparent that if all the sentences for a given environment that can be recognized were contained in a single grammar file then recognition accuracy would be deteriorated and the loading time alone for such grammar and dictionary files would impair the speed of the speech recognition process.

To avoid these problems, specific grammars are dynamically loaded or actively configured as the current grammar according to the user's context, i.e., as in the case of a remote learning system, the Course, Chapter and/or Section selected. Thus the grammar and dictionary files are loaded dynamically according to the given Course, Chapter and/or Section as dictated by the user, or as determined automatically by an application program executed by the user.

The second code block **602** implements the initialization of Speech Recognition engine **182** (FIG. **1**). The MFCC vectors received from client side system **150** along with the grammar filename and the dictionary file names are introduced to this block to initialize the speech decoder.

As illustrated in FIG. **4**A, the initialization process **602** uses the following sub-routines: A routine **602***a* for loading an SRE library. This then allows the creation of an object identified as External Source with code **602***b* using the received MFCC vectors. Code **602***c* allocates memory to hold the recognition objects. Routine **602***d* then also creates and initializes objects that are required for the recognition such as: Source, Coder, Recognizer and Results Loading of the Dictionary created by code **602***e*, Hidden Markov Models (HMMs) generated with code **602***f*; and Loading of the Grammar file generated by routine $\mathbf{602}_g$.

Speech Recognition **603** is the next routine invoked as illustrated in FIG. **4**A, and is generally responsible for completing the processing of the user speech signals input on the client side **150**, which, as mentioned above, are preferably only partially processed (i.e., only MFCC vectors are computed during the first phase) when they are transmitted across link **160**. Using the functions created in External Source by subroutine **602***b*, this code reads MFCC vectors, one at a time from an External Source **603***a*, and processes them in block **603***b* to realize the words in the speech pattern that arc symbolized by the MFCC vectors captured at the client. During this second phase, an additional 13 delta coefficients and an additional 13 acceleration coefficients are computed as part of the recognition process to obtain a total of 39 observation vectors O_t referred to earlier. Then, using a set of previously defined Hidden Markov Models (HMMs), the words corresponding to the user's speech utterance are determined in the manner described earlier. This completes the word "recognition" aspect of the query processing, which results are used further below to complete the query processing operations.

It well be appreciated by those skilled in the art that the distributed nature and rapid performance of the word recognition process, by itself, is extremely useful and may be implemented in connection with other environments that do not implicate or require additional query processing operations. For example, some applications may simply use individual recognized words for filling in data items on a computer generated form, and the aforementioned systems and processes can provide a rapid, reliable mechanism for doing so.

Once the user's speech is recognized, the flow of SRE **182** passes to Un-initialize SRE routine **604** where the speech engine is un-initialized as illustrated. In this block all the objects created in the initialization block are deleted by routine **604***a*, and memory allocated in the initialization block during the initialization phase are removed by routine **604***b*.

Again, it should be emphasized that the above are merely illustrative of embodiments for implementing the particular

routines used on a server side speech recognition system of the present invention. Other variations of the same that achieve the desired functionality and objectives of the present invention will be apparent from the present teachings.

Database Processor **186** Operation—DBProcess

Construction of an SQL Query used as part of the user query processing is illustrated in FIG. **4**B, a SELECT SQL statement is preferably constructed using a conventional CONTAINS predicate. Module **950** constructs the SQL query based on this SELECT SQL statement, which query is used for retrieving the best suitable question stored in the database corresponding to the user's articulated query, (designated as Question here). A routine **951** then concatenates a table name with the constructed SELECT statement. Next, the number of words present in each Noun Phrase of Question asked by the user is calculated by routine **952**. Then memory is allocated by routine **953** as needed to accommodate all the words present in the NP. Next a word List (identifying all the distinct words present in the NP) is obtained by routine **954**. After this, this set of distinct words arc concatenated by routine **955** to the SQL Query separated with a NEAR ( ) keyword. Next, the AND keyword is concatenated to the SQL Query by routine **956** after each NP. Finally memory resources are freed by code **957** so as to allocate memory to store the words received from NP for any next iteration. Thus, at the end of this process, a completed SQL Query corresponding to the user's articulated question is generated.

Connection to SQL Server—As illustrated in FIG. **4**C, after the SQL Query is constructed by routine **710**, a routine **711** implements a connection to the query database **717** to continue processing of the user query. The connection sequence and the subsequent retrieved record set is implemented using routines **700** which include the following:

1. Server and database names are assigned by routine **711**A to a DBProcess member variable
2. A connection string is established by routine **711**B;
3. The SQL Server database is connected under control of code **711**C
4. The SQL Query is received by routine **712**A
5. The SQL Query is executed by code **712**B
6. Extract the total number of records retrieved by the query—**713**
7. Allocate the memory to store the total number of paired questions—**713**
8. Store the entire number of paired questions into an array—**713**

Once the Best Answer ID is received at **716** FIG. **4**C, from the NLE **14** (FIG. **5**), the code corresponding **716**C receives it passes it to code in **716**B where the path of the Answer file is determined using the record number. Then the file is opened **716**C using the path passed to it and the contents of the file corresponding to the answer is read. Then the answer is compressed by code in **716**D and prepared for transmission over the communication channel **160**B (FIG. **1**).

NLQS Database **188**—Table Organization

FIG. **6** illustrates a preferred embodiment of a logical structure of tables used in a typical NLQS database **188** (FIG. **1**). When NLQS database **188** is used as part of NLQS query system **100** implemented as a remote learning/training environment, this database will include an organizational multi-level hierarchy that consists typically of a Course **701**, which is made of several chapters **702**, **703**, **704**. Each of these chapters can have one or more Sections **705**, **706**, **707** as shown for Chapter **1**. A similar structure can exist for Chapter **2**, Chapter **3** . . . Chapter N. Each section has a set of one or more question—answer pairs **708** stored in tables described in more detail below. While this is an appropriate and preferable arrangement for a training/learning application, it is apparent that other implementations would be possible and perhaps more suitable for other applications such as e-commerce, e-support, INTERNET browsing, etc., depending on overall system parameters.

It can be seen that the NLQS database **188** organization is intricately linked to the switched grammar architecture described earlier. In other words, the context (or environment) experienced by the user can be determined at any moment in time based at the selection made at the section level, so that only a limited subset of question-answer pairs **708** for example are appropriate for section **705**. This in turn means that only a particular appropriate grammar for such question-answer pairs may be switched in for handling user queries while the user is experiencing such context. In a similar fashion, an e-commerce application for an INTERNET based business may consist of a hierarchy that includes a first level "home" page **701** identifying user selectable options (product types, services, contact information, etc.), a second level may include one or more "product types" pages **702**, **703**, **704**, a third page may include particular product models **705**, **706**, **707**, etc., and with appropriate question-answer pairs **708** and grammars customized for handling queries for such product models. Again, the particular implementation will vary from application to application, depending on the needs and desires of such business, and a typical amount of routine optimization will be necessary for each such application.

Table Organization

In a preferred embodiment, an independent database is used for each Course. Each database in turn can include three types of tables as follows: a Master Table as illustrated in FIG. **7**A, at least one Chapter Table as illustrated in FIG. **7**B and at least one Section Table as illustrated in FIG. **7**C.

As illustrated in FIG. **7**A, a preferred embodiment of a Master Table has six columns—Field Name **701**A, Data Type **702**A, Size **703**A, Null **704**A, Primary Key **705**A and Indexed **706**A. These parameters are well-known in the art of database design and structure. The Master Table has only two fields—Chapter Name **707**A and Section Name **708**A. Both ChapterName and Section Name are commonly indexed.

A preferred embodiment of a Chapter Table is illustrated in FIG. **7**B. As with the Master Table, the Chapter Table has six (6) columns—Field Name **720**, Data Type **721**, Size **722**, Null **723**, Primary Key **724** and Indexed **725**. There are nine (9) rows of data however, in this case,—Chapter_ID **726**, Answer_ID **727**, Section Name **728**, Answer_Tide **729**, PairedQuestion **730**, AnswerPath **731**, Creator **732**, Date of Creation **733** and Date of Modification **734**.

An explanation of the Chapter Table fields is provided in FIG. **7**C. Each of the eight (8) Fields **720** has a description **735** and stores data corresponding to:

AnswerID **727**—an integer that is automatically incremented for each answer given for user convenience

Section_Name **728**—the name of the section to which the particular record belongs. This field along with the AnswerID is used as the primary key

Answer_Title **729**—A short description of the title of the answer to the user query

PairedQuestion **730**—Contains one or more combinations of questions for the related answers whose path is stored in the next column AnswerPath

AnswerPath **731**—contains the path of a file, which contains the answer to the related questions stored in the previous column; in the case of a pure question/answer application, this file is a text file, but, as mentioned above, could be a multi-media file of any kind transportable over the data link **160**

Creator **732**—Name of Content Creator

Date_of_Creation **733**—Date on which content was created

Date of Modification **734**—Date on which content was changed or modified

A preferred embodiment of a Section Table is illustrated in FIG. **7**D. The Section Table has six (6) columns—Field Name **740**, Data Type **741**, Size **742**, Null **743**, Primary Key **744** and Indexed **745**. There are seven (7) rows of data—Answer_ID **746**, Answer_Title **747**, PairedQuestion **748**, AnswerPath **749**, Creator **750**, Date of Creation **751** and Date of Modification **752**. These names correspond to the same fields, columns already described above for the Master Table and Chapter Table.

Again, this is a preferred approach for the specific type of learning/training application described herein. Since the number of potential applications for the present invention is quite large, and each application can be customized, it is expected that other applications (including other learning/training applications) will require and/or be better accommodated by another table, column, and field structure/hierarchy.

Search Service and Search Engine—A query text search service is performed by an SQL Search System **1000** shown in FIG. **10**. This system provides querying support to process full-text searches. This is where full-text indexes reside.

In general, SQL Search System determines which entries in a database index meet selection criteria specified by a particular text query that is constructed in accordance with an articulated user speech utterance. The Index Engine **1011**B is the entity that populates the Full-Text Index tables with indexes which correspond to the indexable units of text for the stored questions and corresponding answers. It scans through character strings, determines word boundaries, removes all noise words and then populates the full-text index with the remaining words. For each entry in the full text database that meets the selection criteria, a unique key column value and a ranking value are returned as well. Catalog set **1013** is a file-system directory that is accessible only by an Administrator and Search Service **1010**. Full-text indexes **1014** are organized into full-text catalogs, which are referenced by easy to handle names. Typically, full-text index data for an entire database is placed into a single full-text catalog.

The schema for the full-text database as described (FIG. **7**, FIG. **7**A, FIG. **7**B, FIG. **7**C, FIG. **7**D) is stored in the tables **1006** shown in FIG. **10**. Take for example, the tables required to describe the structure the stored question/answer pairs required for a particular course. For each table—Course Table, Chapter Table, Section Table, there are fields—column information that define each parameters that make up the logical structure of the table. This information is stored in User and System tables **1006**. The key values corresponding to those tables are stored as Full-Text catalogs **1013**. So when processing a full-text search, the search engine returns to the SQL Server the key values of the rows that match the search criteria. The relational engine then uses this information to respond to the query.

As illustrated in FIG. **10**, a Full-Text Query Process is implemented as follows:

1. A query **1001** that uses a SQL full-text construct generated by DB processor **186** is submitted to SQL Relational Engine **1002**.
2. Queries containing either a CONTAINS or FREETEXT predicate are rewritten by routine **1003** so that a responsive rowset returned later from Full-Text Provider **1007** will be automatically joined to the table that the predicate is acting upon. This rewrite is a mechanism used to ensure that these predicates are a seamless extension to a traditional SQL Server. After the compiled query is internally rewritten and checked for correctness in item **1003**, the query is passed to RUN TIME module **1004**. The function of module **1004** is to convert the rewritten SQL construct to a validated run-time process before it is sent to the Full-Text Provider, **1007**.
3. After this, Full-Text Provider **1007** is invoked, passing the following information for the query:
   a. A ft_search_condition parameter (this is a logical flag indicating a full text search condition)
   b. A name of a full-text catalog where a full-text index of a table resides
   c. A locale ID to be used for language (for example, word breaking)
   d. Identities of a database, table, and column to be used in the query
   e. If the query is comprised of more than one full-text construct; when this is the case Full-text provider **1007** is invoked separately for each construct.
4. SQL Relational Engine **1002** does not examine the contents of ft_search_condition. Instead, this information is passed along to Full-text provider **1007**, which verifies the validity of the query and then creates an appropriate internal representation of the full-text search condition.
5. The query request/command **1008** is then passed to Querying Support **1011**A.
6. Querying Support **1012** returns a rowset **1009** from Full-Text Catalog **1013** that contains unique key column values for any rows that match the full-text search criteria. A rank value also is returned for each row.
7. The rowset of key column values **1009** is passed to SQL Relational Engine **1002**. If processing of the query implicates either a CONTAINSTABLE( ) or FREETEXTTABLE( ) function, RANK values are returned; otherwise, any rank value is filtered out.
8. The rowset values **1009** are plugged into the initial query with values obtained from relational database **1006**, and a result set **1015** is then returned for further processing to yield a response to the user.

At this stage of the query recognition process, the speech utterance by the user has already been rapidly converted into a carefully crafted text query, and this text query has been initially processed so that an initial matching set of results can be further evaluated for a final determination of the appropriate matching question/answer pair. The underlying principle that makes this possible is the presence of a full-text unique key column for each table that is registered for full-text searches. Thus when processing a full-text search, SQL Search Service **1010** returns to SQL server **1002** the key values of the rows that match the database. In maintaining these full-text databases **1013** and full text indexes **1014**, the present invention has the unique characteristic that the full-text indices **1014** are not updated instantly when the full-text registered columns are updated.

This operation is eliminated, again, to reduce recognition latency, increase response speed, etc. Thus, as compared to other database architectures, this updating of the full-text index tables, which would otherwise take a significant time, is instead done asynchronously at a more convenient time.

Interface Between NLE **190** and DB Processor **188**

The result set **1015** of candidate questions corresponding to the user query utterance are presented to NLE **190** for further processing as shown in FIG. **4**D to determine a "best" matching question/answer pair. An NLE/DBProcessor interface module coordinates the handling of user queries, analysis of noun-phrases (NPs) of retrieved questions sets from the SQL query based on the user query, comparing the retrieved question NPs with the user query NP, etc. between NLE **190** and DB Processor **188**. So, this part of the server side code contains functions, which interface processes resident in both NLE block **190** and DB Processor block **188**. The functions are illustrated in FIG. **4**D; As seen here, code routine **880** implements functions to extract the Noun Phrase (NP) list from the user's question. This part of the code interacts with NLE **190** and gets the list of Noun Phrases in a sentence articulated by the user. Similarly, Routine **813** retrieves an NP list from the list of corresponding candidate/paired questions **1015** and stores these questions into an (ranked by NP value) array. Thus, at this point, NP data has been generated for the user query, as well as for the candidate questions **1015**. As an example of determining the noun phrases of a sentence such as: "What issues have guided the President in considering the impact of foreign trade policy on American businesses?" NLE **190** would return the following as noun phrases: President, issues, impact of foreign trade policy, American businesses, impact, impact of foreign trade, foreign trade, foreign trade policy, trade, trade policy, policy, businesses. The methodology used by NLE **190** will thus be apparent to those skilled in the art from this set of noun phrases and noun sub-phrases generated in response to the example query.

Next, a function identified as Get Best Answer ID **815** is implemented. This part of the code gets a best answer ID corresponding to the user's query. To do this, routines **813**A, **813**B first find out the number of Noun phrases for each entry in the retrieved set **1015** that match with the Noun phrases in the user's query. Then routine **815***a* selects a final result record from the candidate retrieved set **1015** that contains the maximum number of matching Noun phrases.

Conventionally, nouns are commonly thought of as "naming" words, and specifically as the names of "people, places, or things". Nouns such as John, London, and computer certainly fit this description, but the types of words classified by the present invention as nouns is much broader than this. Nouns can also denote abstract and intangible concepts such as birth, happiness, evolution, technology, management, imagination, revenge, politics, hope, cookery, sport, and literacy. Because of the enormous diversity of nouns compared to other parts of speech, the Applicant has found that it is much more relevant to consider the noun phrase as a key linguistic metric. So, the great variety of items classified as nouns by the present invention helps to discriminate and identify individual speech utterances much easier and faster than prior techniques disclosed in the art.

Following this same thought, the present invention also adopts and implements another linguistic entity—the word phrase—to facilitate speech query recognition. The basic structure of a word phrase—whether it be a noun phrase, verb phrase, adjective phrase—is three parts—[pre-Head string],[Head] and [post-Head string]. For example, in the minimal noun phrase—"the children," "children" is classified as the Head of the noun phrase. In summary, because of the diversity and frequency of noun phrases, the choice of noun phrase as the metric by which stored answer is linguistically chosen, has a solid justification in applying this technique to the English natural language as well as other natural languages. So, in sum, the total noun phrases in a speech utterance taken together operate extremely well as unique type of speech query fingerprint.

The ID corresponding to the best answer corresponding to the selected final result record question is then generated by routine **815** which then returns it to DB Process shown in FIG.**4**C. As seen there, a Best Answer ID I is received by routine **716**A, and used by a routine **716**B to retrieve an answer file path. Routine **716**C then opens and reads the answer file, and communicates the substance of the same to routine **716**D. The latter then compresses the answer file data, and sends it over data link **160** to client side system **150** for processing as noted earlier (i.e., to be rendered into audible feedback, visual text/graphics, etc.). Again, in the context of a learning/instructional application, the answer file may consist solely of a single text phrase, but in other applications the substance and format will be tailored to a specific question in an appropriate fashion. For instance, an "answer" may consist of a list of multiple entries corresponding to a list of responsive category items (i.e., a list of books to a particular author) etc. Other variations will be apparent depending on the particular environment.

Natural Language Engine **190**

Again referring to FIG. **4**D, the general structure of NL engine **190** is depicted. This engine implements the word analysis or morphological analysis of words that make up the user's query, as well as phrase analysis of phrases extracted from the query.

As illustrated in FIG. **9**, the functions used in a morphological analysis include tokenizers **802**A, stemmers **804**A and morphological analyzers **806**A. The functions that comprise the phrase analysis include tokenizers, taggers and groupers, and their relationship is shown in FIG.**8**.

Tokenizer **802**A is a software module that functions to break up text of an input sentence **801**A into a list of tokens **803**A. In performing this function, tokenizer **802**A goes through input text **801**A and treats it as a series of tokens or useful meaningful units that are typically larger than individual characters, but smaller than phrases and sentences. These tokens **803**A can include words, separable parts of word and punctuation. Each token **803**A is given an offset and a length. The first phase of tokenization is segmentation, which extracts the individual tokens from the input text and keeps track of the offset where each token originated from in the input text. Next, categories are associated with each token, based on its shape. The process of tokenization is well-known in the art, so it can be performed by any convenient application suitable for the present invention.

Following tokenization, a stemmer process **804**A is executed, which can include two separate forms—inflectional and derivational, for analyzing the tokens to determine their respective stems **805**A. An inflectional stemmer recognizes affixes and returns the word which is the stem. A derivational stemmer on the other hand recognizes derivational affixes and returns the root word or words. While stemmer **804**A associates an input word with its stem, it does not have parts of speech information. Analyzer **806**B takes a word independent of context, and returns a set of possible parts of speech **806**A.

As illustrated in FIG. **8**, phrase analysis **800** is the next step that is performed after tokenization. A tokenizer **802** generates tokens from input text **801**. Tokens **803** are assigned to parts of a speech tag by a tagger routine **804**, and a grouper routine **806** recognizes groups of words as phrases of a certain syntactic type. These syntactic types include for example the noun phrases mentioned earlier, but could include other types if desired such as verb phrases and adjective phrases. Specifically, tagger **804** is a parts-of-speech disambiguator, which analyzes words in context. It has a built-in morphological analyzer (not shown) that allows it to identify all possible parts of speech for each token. The output of tagger **804** is a string with each token tagged with a parts-of-speech label **805**. The final step in the linguistic process **800** is the grouping of words to form phrases **807**. This function is performed by the grouper **806**, and is very dependent, of course, on the performance and output of tagger component **804**.

Accordingly, at the end of linguistic processing **800**, a list of noun phrases (NP) **807** is generated in accordance with the user's query utterance. This set of NPs generated by NLE **190** helps significantly to refine the search for the best answer, so that a single-best answer can be later provided for the user's question.

The particular components of NLE **190** are shown in FIG. **4**D, and include several components. Each of these components implement the several different functions required in NLE **190** as now explained.

Initialize Grouper Resources Object and the Library **900**—this routine initializes the structure variables required to create grouper resource object and library. Specifically, it initializes a particular natural language used by NLE **190** to create a Noun Phrase, for example the English natural language is initialized for a system that serves the English language market. In turn, it also creates the objects (routines) required for Tokenizer, Tagger and Grouper (discussed above) with routines **900**A, **900**B, **900**C and **900**D respectively, and initializes these objects with appropriate values. It also allocates memory to store all the recognized Noun Phrases for the retrieved question pairs.

Tokenizing of the words from the given text (from the query or the paired questions) is performed with routine **909**B—here all the words are tokenized with the help of a local dictionary used by NLE **190** resources. The resultant tokenized words are passed to a Tagger routine **909**C. At routine **909**C, tagging of all the tokens is done and the output is passed to a Grouper routine **909**D.

The Grouping of all tagged token to form NP list is implemented by routine **909**D so that the Grouper groups all the tagged token words and outputs the Noun Phrases.

Un-initializing of the grouper resources object and freeing of the resources, is performed by routines **909**EA, **909**EB and **909**EC. These include Token Resources, Tagger Resources and Grouper Resources respectively. After initialization, the resources are freed. The memory that was used to store all Noun Phrases are also de-allocated.

Additional Embodiments

In a e-commerce embodiment of the present invention as illustrated in FIG. **13**, a web page **1300** contains typical visible links such as Books **1310**, Music **1320** so that on clicking the appropriate link the customer is taken to those pages. The web page may be implemented using HTML, a Java applet, or similar coding techniques which interact with the user's browser. For example, if customer wants to buy an album C by Artist Albert, he traverses several web pages as follows: he first clicks on Music (FIG. **13**, **1360**), which brings up page **1400** where he/she then clicks on Records (FIG. **14**, **1450**). Alternatively, he/she could select CDs **1460**, Videos **1470**, or other categories of books **1410**, music **1420** or help **1430**. As illustrated in FIG. **15**, this brings up another web page **1500** with links for Records **1550**, with sub-categories—Artist **1560**, Song **1570**, Tide **1580**, Genre **1590**. The customer must then click on Artist **1560** to select the artist of choice. This displays another web page **1600** as illustrated in FIG. **16**. On this page the various artists **1650** are listed as illustrated—Albert **1650**, Brooks **1660**, Charlie **1670**, Whyte **1690** are listed under the category Artists **1650**. The customer must now click on Albert **1660** to view the albums available for Albert. When this is done, another web page is displayed as shown in FIG. **17**. Again this web page **1700** displays a similar look and feel, but with the albums available **1760**, **1770**, **1780** listed under the heading Titles **1750**. The customer can also read additional information **1790** for each album. This album information is similar to the liner notes of a shrink-wrapped album purchased at a retail store. One Album A is identified, the customer must click on the Album A **1760**. This typically brings up another text box with the information about its availability, price, shipping and handling charges etc.

When web page **1300** is provided with functionality of a NLQS of the type described above, the web page interacts with the client side and server side speech recognition modules described above. In this case, the user initiates an inquiry by simply clicking on a button designated Contact Me for Help **1480** (this can be a link button on the screen, or a key on the keyboard for example) and is then told by character **1440** about how to elicit the information required. If the user wants Album A by artist Albert, the user could articulate "Is Album A by Brooks available?" in much the same way they would ask the question of a human clerk at a brick and mortar facility. Because of the rapid recognition performance of the present invention, the user's query would be answered in real-time by character **1440** speaking out the answer in the user's native language. If desired, a readable word balloon **1490** could also be displayed to see the character's answer and so that save/print options can also be implemented. Similar appropriate question/answer pairs for each page of the website can be constructed in accordance with the present teachings, so that the customer is provided with an environment that emulates a normal conversational human-like question and answer dialog for all aspects of the web site. Character **1440** can be adjusted and tailored according to the particular commercial application, or by the user's own preferences, etc. to have a particular voice style (man, woman, young, old, etc.) to enhance the customer's experience.

In a similar fashion, an articulated user query might be received as part of a conventional search engine query, to locate information of interest on the INTERNET in a similar manner as done with conventional text queries. If a reasonably close question/answer pair is not available at the server side (for instance, if it does not reach a certain confidence level as an appropriate match to the user's question) the user could be presented with the option of increasing the scope so that the query would then be presented simultaneously to one or more different NLEs across a number of servers, to improve the likelihood of finding an appropriate matching question/answer pair. Furthermore, if desired, more than one "match" could be found, in the same fashion that conventional search engines can return a:number of potential "hits" corresponding to the user's query. For some such queries, of course, it is likely that real-time performance will not be

possible (because of the disseminated and distributed processing) but the advantage presented by extensive supplemental question/answer database systems may be desirable for some users.

It is apparent as well that the NLQS of the present invention is very natural and saves much time for the user and the e-commerce operator as well. In an e-support embodiment, the customer can retrieve information quickly and efficiently, and without need for a live customer agent. For example, at a consumer computer system vendor related support site, a simple diagnostic page might be presented for the user, along with a visible support character to assist him/her. The user could then select items from a "symptoms" page (i.e., a "monitor" problem, a "keyboard" problem, a "printer" problem, etc.) simply by articulating such symptoms in response to prompting from the support character. Thereafter, the system will direct the user on a real-time basis to more specific sub-menus, potential solutions, etc. for the particular recognized complaint. The use of a programmable character thus allows the web site to be scaled to accommodate a large number of hits or customers without any corresponding need to increase the number of human resources and its attendant training issues.

As an additional embodiment, the searching for information on a particular web site may be accelerated with the use of the NLQS of the present invention. Additionally, a significant benefit is that the information is provided in a user-friendly manner through the natural interface of speech. The majority of web sites presently employ lists of frequently asked questions which the user typically wades item by item in order to obtain an answer to a question or issue. For example, as displayed in FIG. **13**, the customer clicks on Help **1330** to initiate the interface with a set of lists. Other options include computer related items at **1370** and frequently asked questions (FAQ) at **1380**.

As illustrated in FIG. **18**, a web site plan for typical web page is displayed. This illustrates the number of pages that have to be traversed in order to reach the list of Frequently-Asked Questions. Once at this page, the user has to scroll and manually identify the question that matches his/her query. This process is typically a laborious task and may or may not yield the information that answers the user's query. The present art for displaying this information is illustrated in FIG. **18**. This figure identifies how the information on a typical web site is organized: the Help link (FIG. **13**, **1330**) typically shown on the home page of the web page is illustrated shown on FIG. **18** as **1800**. Again referring to FIG. **18**, each sub-category of information is listed on a separate page. For example, **1810** lists sub-topics such as 'First Time Visitors', 'Search Tips', 'Ordering', 'Shipping', 'Your Account' etc. Other pages deal with 'Account information' **1860**, 'Rates and Policies' **1850** etc. Down another level, there are pages that deal exclusively with a sub-sub topics on a specific page such as 'First Time Visitors' **1960**, 'Frequently Asked Questions' **1950**, 'Safe Shopping Guarantee' **1940**, etc. So if a customer has a query that is best answered by going to the Frequently Asked Questions link, he or she has to traverse three levels of busy and cluttered screen pages to get to the Frequently Asked Questions page **1950**. Typically, there are many lists of questions **1980** that have to be manually scrolled through. While scrolling visually, the customer then has to visually and mentally match his or her question with each listed question. If a possible match is sighted, then that question is clicked and the answer then appears in text form which then is read.

In contrast, the process of obtaining an answer to a question using a web page enabled with the present NLQS can be achieved much less laboriously and efficiently. The user would articulate the word "Help" (FIG. **13**, **1330**). This would immediately cause a character (FIG. **13**, **1340**) to appear with the friendly response "May I be of assistance. Please state your question?". Once the customer states the question, the character would then perform an animation or reply "Thank you, I will be back with the answer soon". After a short period time (preferably not exceeding 5–7 seconds) the character would then speak out the answer to the user's question. As illustrated in FIG. **18** the answer would be the answer **1990** returned to the user in the form of speech is the answer that is paired with the question **1950**. For example, the answer **1990**: "We accept Visa, MasterCard and Discover credit cards", would be the response to the query **2000** "What forms of payments do you accept?"

Another embodiment of the invention is illustrated in FIG. **12**. This web page illustrates a typical website that employs NLQS in a web-based learning environment. As illustrated in FIG. **12**, the web page in browser **1200**, is divided into two or more frames. A character **1210** in the likeness of an instructor is available on the screen and appears when the student initiates the query mode either by speaking the word "Help" into a microphone (FIG. **2**, **215**) or by clicking on the link 'Click to Speak' (FIG. **12**, **1280**). Character **1210** would then prompt the student to select a course **1220** from the drop down list **1230**. If the user selects the course 'CPlusPlus', the character would then confirm verbally that the course "CPlusPlus" was selected. The character would then direct the student to make the next selection from the drop-down list **1250** that contains the selections for the chapters **1240** from which questions are available. Again, after the student makes the selection, the character **1210** confirms the selection by speaking. Next character **1210** prompts the student to select 'Section' **1260** of the chapter from which questions are available from the drop down list **1270**. Again, after the student makes the selection, character **1210** confirms the selection by articulating the 'Section' **1260** chosen. As a prompt to the student, a list of possible questions appear in the list box **1291**. In addition, tips **1290** for using the system are displayed. Once the selections are all made, the student is prompted by the character to ask the question as follows: "Please ask your query now". The student then speaks his query and after a short period of time, the character responds with the answer preceded by the question as follows: "The answer to your question . . . is as follows: . . . ". This procedure allows the student to quickly retrieve answers to questions about any section of the course and replaces the tedium of consulting books, and references or indices. In short, it is can serve a number of uses from being a virtual teacher answering questions on-the-fly or a flash card substitute.

From preliminary data available to the inventors, it is estimate that the system can easily accommodate 100–250 question/answer pairs while still achieving a real-time feel and appearance to the user (i.e., less than 10 seconds of latency, not counting transmission) using the above described structures and methods. It is expected, of course, that these figures will improve as additional processing speed becomes available, and routine optimizations are employed to the various components noted for each particular environment.

Again, the above are merely illustrative of the many possible applications of the present invention, and it is expected that many more web-based enterprises, as well as other consumer applications (such as intelligent, interactive toys) can utilize the present teachings. Although the present invention has been described in terms of a preferred

embodiment, it will be apparent to those skilled in the art that many alterations and modifications may be made to such embodiments without departing from the teachings of the present invention. It will also be apparent to those skilled in the art that many aspects of the present discussion have been simplified to give appropriate weight and focus to the more germane aspects of the present invention. The microcode and software routines executed to effectuate the inventive methods may be embodied in various forms, including in a permanent magnetic media, a non-volatile ROM, a CD-ROM, or any other suitable machine-readable format. Accordingly, it is intended that the all such alterations and modifications be included within the scope and spirit of the invention as defined by the following claims.

What is claimed is:

1. An interactive learning system adapted for responding to speech-based queries concerning topics addressed by such interactive learning system, the system comprising:

a query file for storing a plurality of topic query entries, each topic query entry including a query relating to one or more of the topics covered by the speech-based interactive learning system; and

an answer file for storing a plurality of topic answer entries, each topic answer entry including an answer to one or more of said plurality of topic query entries, such that each topic query entry has at least one associated topic answer entry; and

a speech recognition system for generating recognized speech utterance data from partially processed speech data associated with a speech-based query concerning one of said topics, said partially processed speech data being received from a remote speech capturing system; and

said speech recognition system further cooperating with a natural language engine, which processes said recognized speech utterance data using a morphological analysis and a phrase analysis to form recognized speech sentence data corresponding to said speech-based query;

a query formulation system for converting said recognized speech sentence data into a search query suitable for identifying a topic query entry corresponding to said speech-based query, and for locating at least one topic answer entry best matching said speech-based query.

2. The system of claim **1**, wherein said remote speech capturing system is located at a client site, and said speech recognition system is distributed across said client site and a separate server site.

3. The system of claim **1**, wherein said speech recognition system is comprised of a first portion at a client based computing system for performing first signal processing operations on a speech input signal to create said partially processed speech data, and a second portion at a server based computing system for performing a second signal processing operation for completing processing of said partially processed speech data.

4. The system of claim **1**, wherein said query formulation system uses context parameters for recognizing said speech-based query.

5. The system of claim **4**, wherein said context parameters include any one or more of the following: a course, chapter, and/or section of said lesson information selected by said user during the interactive session.

6. The system of claim **4**, wherein said context parameters include any one or more visible text words or objects presented to said user during the interactive session.

7. The system of claim **4**, wherein said context parameters are used for dynamically determining and loading an appropriate grammar and dictionary file to be used for said speech-based query.

8. A speech based interactive learning system comprising:

an instructional file containing instructional materials arranged in a hierarchical structure that includes at least a first level of instruction data and a second level of instruction data; and

wherein users of such system can navigate said hierarchical structure and formulate a user query concerning instructional materials located at least at said second level of instruction data;

further wherein said user query can be correlated with a corresponding instructional material question taken from a list of predefined questions, said set of predefined questions being paired with a corresponding set of responsive answers;

a speech recognition engine for generating recognized words from a user query;

a natural language engine for parsing said words contained in said user query to generate recognized speech sentence data;

a query formulation engine for converting said recognized speech sentence data into a search query suitable for identifying a corresponding instructional material question for said user query, and for locating a corresponding responsive answer for said corresponding instructional material question.

9. A speech based interactive learning system comprising:

an instructional file containing instructional materials arranged in a hierarchical structure that includes at least a first level of instruction data and a second level of instruction data; and

wherein users of such system can navigate said hierarchical structure and formulate a user query concerning instructional materials located at least at said second level of instruction data;

further wherein said user query can be correlated with a corresponding instructional material question taken from a list of predefined questions, said set of predefined questions being paired with a corresponding set of responsive answers;

a speech recognition engine for generating recognized speech data from a user query;

a query formulation engine for converting said recognized speech data into a search query suitable for identifying a corresponding instructional material question for said user query, and for locating a corresponding responsive answer for said corresponding instructional material question;

wherein said search query is formulated using a natural language engine parsing words contained in said user query.

10. The system of claim **9**, wherein said natural language engine compares said user query with one or more of said list of predefined questions to determine said corresponding instructional material question.

11. The system of claim **8**, where a set of potential questions corresponding to said user query is derived from said list of predefined questions by partially recognizing said user query, and said corresponding instructional material question is derived by fully recognizing said user query from said set of potential questions.

12. The system of claim **9**, wherein said first level of instruction data is associated with one or more lesson

chapters for a particular course, and said second level of instruction data is associated with one or more sections, said one or more sections being linked to said one or more chapters.

13. The system of claim **8**, further including an animated visible agent for assisting said user to navigate said instructional materials, and for articulating said corresponding responsive answer in audible form to said user.

14. The system of claim **9**, further including an animated visible agent for assisting said user to navigate said instructional materials, and for articulating said corresponding responsive answer to said user.

15. A speech based system for assisting a user in connection with an interactive lesson tutorial having question and answer capability, the system comprising:

a lesson file containing instructional materials arranged to include a list of predefined questions and a corresponding list of predefined answers for said lesson; and

a speech recognition engine for generating recognized speech word data from a user query pertaining to said lesson; and

a natural language engine for generating recognized speech sentence data from said speech word data; and

a query recognition engine for locating a corresponding predefined question for said user query using said recognized speech sentence data; and

a conversion engine for converting a corresponding predefined answer for said corresponding predefined question into a form perceptible to the user;

said query recognition system being adapted to process said user query, even before said user has completed articulating said user query, to identify said corresponding predefined answer and thus emulate a human response time in response to a user query, so that the user perceives interaction with such system in essentially the same way that would be experienced from interacting with a real human.

16. A speech based system for assisting a user in connection with an interactive lesson tutorial having question and answer capability, the system comprising:

a lesson file containing instructional materials arranged to include a list of predefined questions and a corresponding list of predefined answers for said lesson; and

a speech recognition engine for generating recognized speech data from a user query pertaining to said lesson; and

a query recognition engine for locating a corresponding predefined question for said user query using said recognized speech data; and

a conversion engine for converting a corresponding predefined answer for said corresponding predefined question into a form perceptible to the user;

said query recognition system being adapted to emulate a human response time in response to a user query, so that the user perceives interaction with such system in essentially the same way that would be experienced from interacting with a real human.

17. The system of claim **16**, wherein said system responds to a user query in less than approximately 10 seconds for a list of predefined questions having at least 100 entries, not accounting for data transmission latencies between the user and the system.

18. The system of claim **16**, wherein said system further includes a visible interactive agent that receives and responds to said user query.

19. The system of claim **16**, wherein said interactive agent is configured with an appearance and mannerism that emulates a corresponding human appearance and mannerism appropriate for said lesson tutorial.

20. The system of claim **15**, wherein said query recognition engine is distributed across multiple computing systems so that said more than one lesson file is consulted for said user query.

21. A method of implementing a speech-based interactive query system, including the steps of:

(a) storing a plurality of topic query entries, each topic query entry including a query relating to one or more of topics covered by the speech-based interactive learning system; and

(b) storing a plurality of topic answer entries, each topic answer entry including an answer to one or more of said plurality of topic query entries, such that each topic query entry has at least one associated topic answer entry; and

(c) generating recognized speech utterance data associated with a speech-based query concerning one of said topics, such that said recognized speech utterance data is generated by partial recognition processing of said speech-based query by a first signal processing routine executing at a first computing device, and then completing recognition of said speech-based query through processing performed by a second signal processing routine executing at a second computing device; and

(d) converting said recognized speech utterance data with a natural language process into recognized speech sentence data, said recognized speech data being used by a search query suitable for identifying a topic query entry corresponding to said speech-based query; and

(e) locating at least one topic answer entry best matching said speech-based query;

wherein step (e) is initiated before said natural language engine has converted said recognized speech utterance data into recognized speech sentence data.

22. A method of implementing a speech-based interactive query system, including the steps of:

(a) storing a plurality of topic query entries, each topic query entry including a query relating to one or more of topics covered by the speech-based interactive learning system; and

(b) storing a plurality of topic answer entries, each topic answer entry including an answer to one or more of said plurality of topic query entries, such that each topic query entry has at least one associated topic answer entry; and

(c) generating recognized speech data associated with a speech-based query concerning one of said topics, such that said recognized speech data is generated by partial recognition processing of said speech-based query by a first signal processing routine executing at a first computing device, and then completing recognition of said speech-based query through processing performed by a second signal processing routine executing at a second computing device; and

(d) converting said recognized speech data into a search query suitable for identifying a topic query entry corresponding to said speech-based query; and

(e) locating at least one topic answer entry best matching said speech-based query.

23. The method of claim **21**, wherein during step (d) context parameters are used for formulating said search

query, and said context parameters are used for dynamically determining and loading an appropriate grammar and dictionary file to be used for said speech-based query.

24. The method of claim **1**, wherein during step (d) context parameters are used for formulating said search query, and said context parameters are used for dynamically determining and loading an appropriate grammar and dictionary file to be used for said speech-based query.

25. A method of presenting an interactive lesson to a user comprising the steps of:

(a) arranging instructional materials in a hierarchical structure that includes at least a first level of instruction data and a second level of instruction data; and

(b) presenting the instructional materials to a user so that the user can navigate said hierarchical structure and formulate a user query concerning instructional materials located at least at said second level of instruction data;

(c) defining a list of predefined questions for at least said second level of instruction data, said set of predefined questions being paired with a corresponding set of responsive answers;

(d) generating recognized speech utterance data from a user query for such interactive lesson;

(e) converting said recognized speech utterance data into a search query suitable for identifying a corresponding instructional material question for said user query, said search query being formulated with assistance from a natural language engine parsing words in said recognized speech utterance data; and

(f) identifying a corresponding responsive answer for said corresponding instructional material question;

wherein a set of potential questions corresponding to said user query is derived by partially processing said speech utterance data during step (e), and then said corresponding instructional material question is determined by fully processing said speech utterance data.

26. The method of claim **25**, wherein said response undergoes a text to speech process so that said topic answer entry is expressed in audible form to a user.

27. A method of presenting an interactive lesson to a user comprising the steps of:

(a) arranging instructional materials in a hierarchical structure that includes at least a first level of instruction data and a second level of instruction data; and

(b) presenting the instructional materials to a user so that the user can navigate said hierarchical structure and formulate a user query concerning instructional materials located at least at said second level of instruction data;

(c) defining a list of predefined questions for at least said second level of instruction data, said set of predefined questions being paired with a corresponding set of responsive answers;

(d) generating recognized speech data from a user query for such interactive lesson;

(e) converting said recognized speech data into a search query suitable for identifying a corresponding instructional material question for said user query, said search query being formulated with assistance from a natural language engine parsing words in said recognized speech data; and

(f) identifying a corresponding responsive answer for said corresponding instructional material question.

28. The method of claim **27**, wherein during step (f) said natural language engine operates on said recognized speech data as well as on one or more of said list of predefined questions to determine said corresponding instructional material question.

29. The method of claim **27**, wherein said first level of instruction data is associated with one or more lesson chapters for a particular course, and said second level of instruction data is associated with one or more sections, said one or more sections being linked to said one or more chapters.

30. A method of operating a speech-based lesson tutorial system having question and answer capability, the method comprising the steps of:

(a) configuring a list of retrievable predefined questions and a corresponding list of predefined answers for said lesson; and

(b) generating recognized speech data from a user query pertaining to said lesson; and

wherein said recognized speech data is generated by a combination of both speech utterance recognition and natural language processing;

(c) locating a corresponding predefined question for said user query using said recognized speech data; and

(d) converting a corresponding predefined answer for said corresponding predefined question into a form perceptible to the user; and

wherein at least step (b) is performed at least in part while a user is articulating said query so as to emulate a human response time in response to a user query, so that the user perceives interaction with such system in essentially the same way that would be experienced from interacting with a real human.

31. The method of claim **27**, further including a step (g): displaying an animated visible agent for assisting said user to navigate said instructional materials, and for articulating said corresponding responsive answer to said user.

32. The method of claim **27**, wherein during step (e) context parameters are used for recognizing said speech-based query, and said context parameters are used for dynamically determining and loading an appropriate grammar and dictionary file to be used for said speech-based query.

33. A method of operating a speech-based lesson tutorial system having question and answer capability, the method comprising the steps of:

(a) configuring a list of retrievable predefined questions and a corresponding list of predefined answers for said lesson; and

(b) generating recognized speech data from a user query pertaining to said lesson; and

(c) locating a corresponding predefined question for said user query using said recognized speech data; and

(d) converting a corresponding predefined answer for said corresponding predefined question into a form perceptible to the user; and

wherein steps (b) to (d) are performed so as to emulate a human response time in response to a user query, so that the user perceives interaction with such system in essentially the same way that would be experienced from interacting with a real human.

34. The method of claim **33**, wherein step (d) occurs in response to a user query in less than approximately 10 seconds for a list of predefined questions having at least 100 entries, not accounting for data transmission latencies between the user and the tutorial system.

35. The method of claim **30**, wherein step (c) occurs across multiple computing systems so that said more than one lesson file is consulted for said user query.

36. The method of claim **35**, wherein said interactive agent is configured with an appearance and mannerism that emulates a corresponding human appearance and mannerism appropriate for said lesson tutorial.

37. The method of claim **33**, wherein during step (c) context parameters are used for recognizing said speech-based query, and said context parameters are used for dynamically determining and loading an appropriate grammar and dictionary file to be used for said speech-based query.

38. The method of claim **33**, wherein step (c) occurs across multiple computing system so that said more than one lesson file is consulted for said user query.

* * * * *

EXHIBIT 3

US007050977B1

(12) **United States Patent**
Bennett

(10) **Patent No.: US 7,050,977 B1**
(45) **Date of Patent: May 23, 2006**

(54) **SPEECH-ENABLED SERVER FOR INTERNET WEBSITE AND METHOD**

(75) Inventor: **Ian M. Bennett**, Palo Alto, CA (US)

(73) Assignee: **Phoenix Solutions, Inc.**, Palo Alto, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/439,174**

(22) Filed: **Nov. 12, 1999**

(51) **Int. Cl.**
G10L 15/02 (2006.01)
G10L 15/18 (2006.01)
G10L 15/22 (2006.01)
G06F 17/20 (2006.01)

(52) **U.S. Cl.** **704/270.1**; 704/275; 707/3

(58) **Field of Classification Search** 704/231–235, 704/251, 255, 257, 260, 270, 270.1, 275; 707/3, 4, 5
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,785,408	A	11/1988	Britton et al.	
4,852,170	A	7/1989	Bordeaux	
5,371,901	A	12/1994	Reed et al.	
5,509,104	A	4/1996	Lee et al.	
5,553,119	A	9/1996	McAlliser et al.	
5,652,897	A	7/1997	Linebarger et al.	
5,668,854	A	9/1997	Minakami et al.	
5,675,707	A	10/1997	Gorin et al.	
5,680,511	A	10/1997	Baker et al.	
5,758,322	A	5/1998	Rongley	704/275
5,802,256	A	9/1998	Fawcett et al.	
5,819,220	A	10/1998	Sarukkai et al.	704/243
5,836,771	A	11/1998	Ho et al.	
5,860,063	A	1/1999	Gorin et al.	
5,867,817	A	2/1999	Catallo et al.	704/255
5,873,062	A	2/1999	Hansen et al.	
5,884,302	A	3/1999	Ho	
5,915,236	A	6/1999	Gould et al.	704/251
5,934,910	A	8/1999	Ho et al.	
5,956,683	A	9/1999	Jacobs et al.	704/275
5,960,394	A	9/1999	Gould et al.	704/240
5,960,399	A	9/1999	Barclay et al.	704/270

(Continued)

FOREIGN PATENT DOCUMENTS

EP 1 094 388 A2 4/2001

(Continued)

OTHER PUBLICATIONS

Arons, B., "The Design of Audio Servers and Toolkits for Supporting Speech in the User Interface," believed to be published in: Journal of the American Voice I/O Society, pp. 27-41, Mar. 1991.

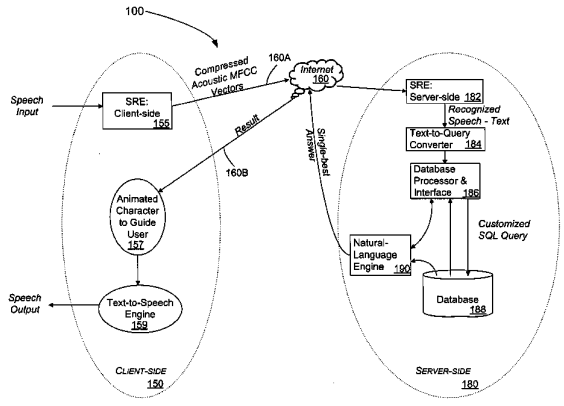
(Continued)

Primary Examiner—Martin Lerner
(74) *Attorney, Agent, or Firm*—J. Nicholas Gross

(57) **ABSTRACT**

An Internet-based server with speech support for enhanced interactivity is disclosed. This server hosts a server-side speech recognition engine and additional linguistic and database functions that cooperate to provide enhanced interactivity for clients so that their browsing experience is more satisfying, efficient and productive. This human-like interactivity which allows the user to ask queries about topics that range from customer delivery, product descriptions, payment details, is facilitated by the allowing the user to articulate the his or her questions directly in his or her natural language. The answer typically provided in real-time, can also be interfaced and integrated with existing telephone, e-mail and other mixed media services to provide a single point of interactivity for the user when browsing at a web-site.

78 Claims, 31 Drawing Sheets



U.S. PATENT DOCUMENTS

5,978,756	A		11/1999	Walker et al.
5,987,410	A		11/1999	Kellner et al.
5,995,918	A		11/1999	Kendall et al.
5,995,928	A		11/1999	Nguyen et al.
6,009,387	A		12/1999	Ramaswamy et al.
6,029,124	A		2/2000	Gillick et al. 704/200
6,032,111	A		2/2000	Mohri
6,035,275	A		3/2000	Brode et al.
6,044,266	A		3/2000	Kato
6,044,337	A		3/2000	Gorin et al.
6,101,472	A	*	8/2000	Giangarra et al. 704/275
6,112,176	A		8/2000	Goldenthal et al.
6,119,087	A		9/2000	Kuhn et al.
6,125,284	A		9/2000	Moore et al.
6,125,341	A		9/2000	Raud et al.
6,138,089	A		10/2000	Guberman 704/207
6,141,640	A		10/2000	Moo
6,144,848	A		11/2000	Walsh et al.
6,144,938	A		11/2000	Surace et al. 704/257
6,178,404	B1		1/2001	Hambleton et al.
6,182,038	B1		1/2001	Balakrishnan et al.
6,182,068	B1		1/2001	Culliss
6,185,535	B1		2/2001	Hedin et al.
6,192,110	B1		2/2001	Abella et al.
6,195,636	B1		2/2001	Crupi et al.
6,226,610	B1		5/2001	Keiller et al.
6,233,559	B1		5/2001	Balakrishnan
6,243,679	B1		6/2001	Mohri et al.
6,246,986	B1		6/2001	Ammicht et al.
6,246,989	B1		6/2001	Polcyn
6,256,607	B1		7/2001	Digalakis et al.
6,269,336	B1		7/2001	Ladd et al.
6,278,973	B1		8/2001	Chung et al.
6,292,767	B1		9/2001	Jackson et al.
6,292,781	B1		9/2001	Urs et al.
6,327,561	B1		12/2001	Smith et al.
6,327,568	B1		12/2001	Joost
6,330,530	B1	*	12/2001	Horiguchi et al. 704/4
6,363,349	B1		3/2002	Urs et al.
6,374,219	B1		4/2002	Jiang
6,374,226	B1		4/2002	Hunt et al.
6,381,594	B1		4/2002	Eichstaedt et al.
6,389,389	B1		5/2002	Meunier et al.
6,408,272	B1		6/2002	White et al.
6,411,926	B1		6/2002	Chang
6,418,199	B1	*	7/2002	Perrone 379/88.01
6,427,063	B1		7/2002	Cook et al.
6,434,529	B1		8/2002	Walker et al.
6,453,020	B1		9/2002	Hughes et al.
6,499,011	B1		12/2002	Souvignier et al.
6,499,013	B1		12/2002	Weber
6,510,411	B1		1/2003	Norton et al.
6,513,037	B1		1/2003	Ruber et al.
6,522,725	B1		2/2003	Kato
6,532,444	B1		3/2003	Weber
6,539,359	B1	*	3/2003	Ladd et al. 704/275
6,567,778	B1		5/2003	Chao Chang et al.
6,574,597	B1		6/2003	Mohri et al.
6,584,464	B1		6/2003	Warthen
6,594,269	B1		7/2003	Polcyn
6,594,348	B1		7/2003	Bjurstrom et al.
6,614,885	B1		9/2003	Polcyn
6,618,726	B1	*	9/2003	Colbath et al. 707/6
6,633,846	B1	*	10/2003	Bennett et al. 704/257
6,681,206	B1		1/2004	Gorin et al.
6,697,780	B1		2/2004	Beutnagel et al.
6,742,021	B1	*	5/2004	Halverson et al. 709/218
6,823,308	B1		11/2004	Keiller et al.
6,871,179	B1		3/2005	Kist et al.
6,901,366	B1	*	5/2005	Kuhn et al. 704/275
6,922,733	B1	*	7/2005	Kuiken et al. 709/246
6,940,953	B1	*	9/2005	Eberle et al. 379/88.13
6,941,273	B1	*	9/2005	Loghmani et al. 705/26
6,961,954	B1	*	11/2005	Maybury et al. 725/53
6,964,012	B1	*	11/2005	Zirngibl et al. 715/513
6,965,864	B1	*	11/2005	Thrift et al. 704/275
6,965,890	B1	*	11/2005	Dey et al. 707/4
2001/0016813	A1		8/2001	Brown et al.
2001/0032083	A1		10/2001	Van Cleven
2001/0056346	A1		12/2001	Ueyama et al.
2002/0032566	A1		3/2002	Tzirkel-Hancock et al.
2002/0046023	A1		4/2002	Fujii et al.
2002/0059068	A1		5/2002	Rose et al.
2002/0059069	A1		5/2002	Hsu et al.
2002/0086269	A1		7/2002	Shpiro
2002/0087325	A1		7/2002	Lee et al.
2002/0087655	A1		7/2002	Bridgman et al.
2002/0091527	A1		7/2002	Shiau
2003/0191625	A1		10/2003	Gorin et al.
2005/0091056	A1		4/2005	Surace et al.
2005/0131704	A1		6/2005	Dragosh et al.

FOREIGN PATENT DOCUMENTS

EP	1 096 471 A1	5/2001
WO	WO 99/48011 A1	9/1999
WO	WO 99/50830	10/1999
WO	WO 00/14727 A1	3/2000
WO	WO 00/17854 A1	3/2000
WO	WO 00/20962 A2	4/2000
WO	WO 00/21075 A1	4/2000
WO	WO 00/21232 A2	4/2000
WO	WO 00/22610 A1	4/2000
WO	WO 00/30072 A2	5/2000
WO	WO 00/30287 A1	5/2000
WO	WO 00/68823 A2	11/2000
WO	WO 01/16936 A1	3/2001
WO	WO 01/18693 A2	3/2001
WO	WO 01/26093 A1	4/2001
WO	WO 01/78065 A1	10/2001
WO	WO 01/95312 A1	12/2001
WO	WO 02/03380 A1	1/2002

OTHER PUBLICATIONS

Hazen, T et al., "Recent Improvements in an Approach to Segment-Based Automatic Language Identification," believed to be published in: Proceedings of the 1994 International Conference on Spoken Language Processing, Yokohama, Japan, pp. 1883-1886, Sep. 1994.

House, D., "Spoken-Language Access to Multimedia (SLAM): A Multimodal Interface to the World-Wide Web," Masters Thesis, Oregon Graduate Institute, Department of Computer Science & Engineering, 59 pages, Apr. 1995.

Julia, L. et al., "http://www.speech. sri.com/demos/ atis. html," believed to be published in: Proceedings AAAI'97: Stanford, pp. 72-76, Jul. 1997.

Lau, R. et al, "Webgalaxy-Integrating Spoken Language and Hypertext Navigation," believed to be published in: in Kokkinakis, G. et al., (Eds.) Eurospeech '97, Proceedings of the 5th European Conference on Speech Communication and Technology, Rhodes (Greece), Sep. 22-25, 1997: pp. 883-886, 1997.

Digalakis, V. et al., "Product-Code Vector Quantization of Cepstral Parameters for Speech Recognition over the WWW," believed to be published in: Proc. ICSLP '98, 4 pages, 1998.

Melin, H., "On Word Boundary Detection in Digit-Based Speaker Verification," believed to be published in: Workshop on Speaker Recognition and Its Commercial and Forensic Applications (RLA2C), Avignon, France, Apr. 20-23, pp. 46-49, 1998.

Ramaswamy, G. et al., "Compression of Acoustic Features for Speech Recognition in Network Environments," believed to be published in: IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 977-980, Jun. 1998.
Lu, B. et al., "Scalability Issues in the Real Time Protocol (RTP)," Project Report for CPSC 663 (Real Time Systems), Dept. of Computer Science, Texas A & M University, 19 pages, 1999.
Giuliani, D. et al., "Training of HMM with Filtered Speech Material for Hands-Free Recognition," believed to be published in: Proceedings of ICASSP '99, Phoenix, USA, 4 pages, 1999.
Digalakis,V. et al., "Quantization of Cepstral Parameters for Speech Recognition over the World Wide Web," believed to be published in: IEEE Journal on Selected Areas of Communications, 22 pages, 1999.
Tsakalidis, S. et al., "Efficient Speech Recognition Using Subvector Quantization and Discrete-Mixture HMMs," believed to be published in: Proc. ICASSP '99, 4 pages, 1999.
Lin, B. et al., "A Distributed Architecture for Cooperative Spoken Dialogue Agents with Coherent Dialogue State and History," believed to be published in: IEEE Automatic Speech Recognition and Understanding Workshop, Keystone, Colorado, USA, 4 pages, Dec. 1999.
Meunier, J., "RTP Payload Format for Distributed Speech Recognition," 48th IETF AVT WG—Aug. 3, 2000, 10 pages, 2000.
Sand Cherry Networks, SoftServer product literature, 2 pages, 2001.
Kim,H. et al., "A Bitstream-Based Front-End for Wireless Speech Recognition on IS-136 Communications System," IEEE Transactions on Speech and Audio Processing, vol. 9, No. 5, pp. 558-568, Jul. 2001.
Agarwal, R., Towards a PURE Spoken Dialogue System for Information Access, believed to be published in *Proceedings of the ACL/EACL Workshop on Interactive Spoken Dialog Systems: Bringing Speech and NLP Together in Real Applications*, Madrid, Spain, 1997, 9 pages.
Ammicht, Egbert et al., "Knowledge Collection for Natural Language Spoken Dialog Systems," believed to be published in *Proc. Eurospeech*, vol. 3, p. 1375-1378, Budapest, Hungary, Sep. 1999, 4 pages.
AT&T Corp., "Network Watson 1.0 System Overview," 1998, 4 pages.
AT&T Corp., "AT&T Watson Advanced Speech Applications Platform," 1996, 3 pages.
AT&T Corp., "AT&T Watson Advanced Speech Applications Platform Version 2.0," 1996, 3 pages.
AT&T Corp., "AT&T Watson Advanced Speech Application Platform Version 2.0," 1996, 8 pages.
Gorin, Allen, "Processing of Semantic Information in Fluently Spoken Language," believed to be published in *Proc. ICSLP*, Philadelphia, PA, Oct. 1996, 4 pages.
Gorin, Allen et al., "How May I Help You," believed to be published in *Proc. IVTTA*, Basking Ridge, NJ, Oct. 1996, 32 pages.
Mohru, Mehryar, "String Matching With Automata," Nordic Journal of Computing, 1997, 15 pages.
Prudential News, "Prudential Pilots Revolutionary New Speech-Based Telephone Customer Service System Developed by AT&T Labs—Company Business and Marketing," Dec. 6, 1999, 3 pages.
Riccardi, Giuseppe et al., "A spoken language system for automated call routing," believed to be published in *Proc. ICASSP '97*, 1997, 4 pages.
Sharp, Douglas, et al., "The Watson Speech Recognition Engine," accepted by ICASSP, 1997, 9 pages.
European Patent Office search report for EP Application No. 00977144, dated Mar. 30, 2005, 5 pages.
Burstein, A. et al. "Using Speech Recognition In A Personal Communications System," *Proceedings of the International Conference on Communications*; Chicago, Illinois, Jun. 14-18, 1992, pp. 1717-1721.
Digalakis, V. et al., "Quantization Of Cepstral Parameters For Speech Recognition Over The World Wide Web," *IEEE Journal on Selected Areas of Communications*, 1999, pp. 82-90.
Kuhn, T. et al., "Hybrid In-Car Speech Recognition For Mobile Multimedia Applications," *Vehicular Technology Conference*, Houston, Texas, May 1999, pp. 2009-2013.
L. Travis, "Handbook of Speech Pathology", Appleton-Century-Crofts, Inc., 1957, pp. 91-12-4.
L.E. Baum, T. Petrie, "Statistical inference for probabilistic functions for finite state Markov chains", Ann. Math. Stat., 37: 1554-1563, 1966.
J.L. Flanagan, "Speech Analysis Synthesis and Perception", 2nd edition, Springer-Verlag Berlin, 1972, pp. 1-53.
L.E. Baum, "An inequality and associated maximation technique in statistical estimation for probabilistic functions of Markov processes", Inequalities 3: 1-8, 1972.
Cox, Richard V. et al., "Speech and Language Processing for Next-Millennium Communications Services," Proceedings of the IEEE, vol. 88, No. 8, Aug. 2000, pp. 1314-1337.
Kuhn, Roland, and Renato De Mori, "The Application of Semantic Classification Trees to Natural Language Understanding," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 17, No. 5, May 1995, pp. 449-460.
Unisys Corp., "Natural Language Speech Assistant (NLSA) Capabilities Overview," NLR 3.0, Aug. 1998, Malvern, PA, 27 pages.

* cited by examiner

Fig. 1

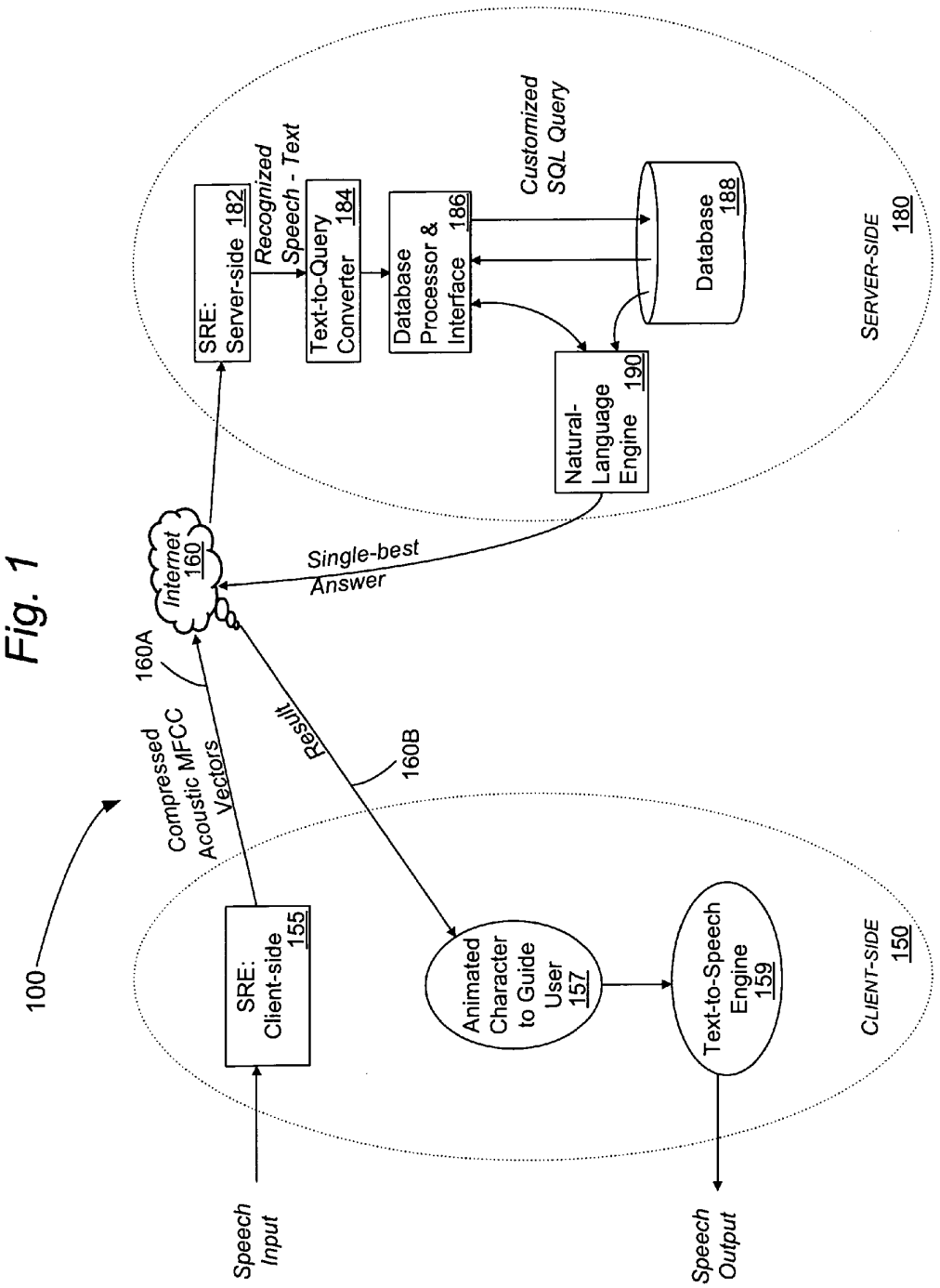

100
SRE: Client-side 155
Speech Input
Compressed MFCC Acoustic Vectors
160A
Internet 160
Result
160B
Single-best Answer
SRE: Server-side 182
Recognized Speech - Text
Text-to-Query Converter 184
Database Processor & Interface 186
Customized SQL Query
Database 188
Natural-Language Engine 190
SERVER-SIDE 180
Animated Character to Guide User 157
Text-to-Speech Engine 159
Speech Output
CLIENT-SIDE 150

Figure 2 (Page 1/3)
CLIENT-SIDE SYSTEM LOGIC

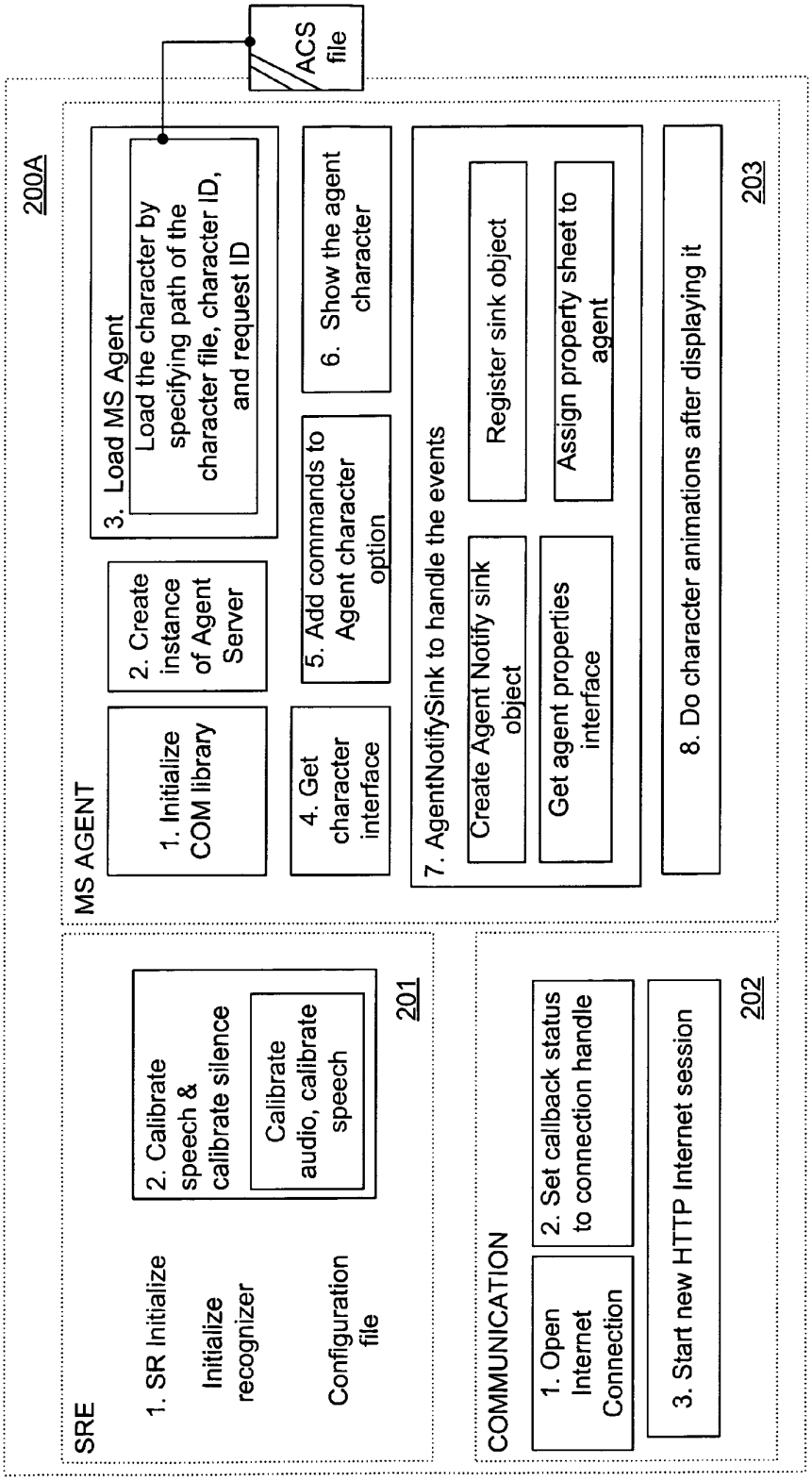


Figure 2 (Page 2/3)
CLIENT-SIDE SYSTEM LOGIC

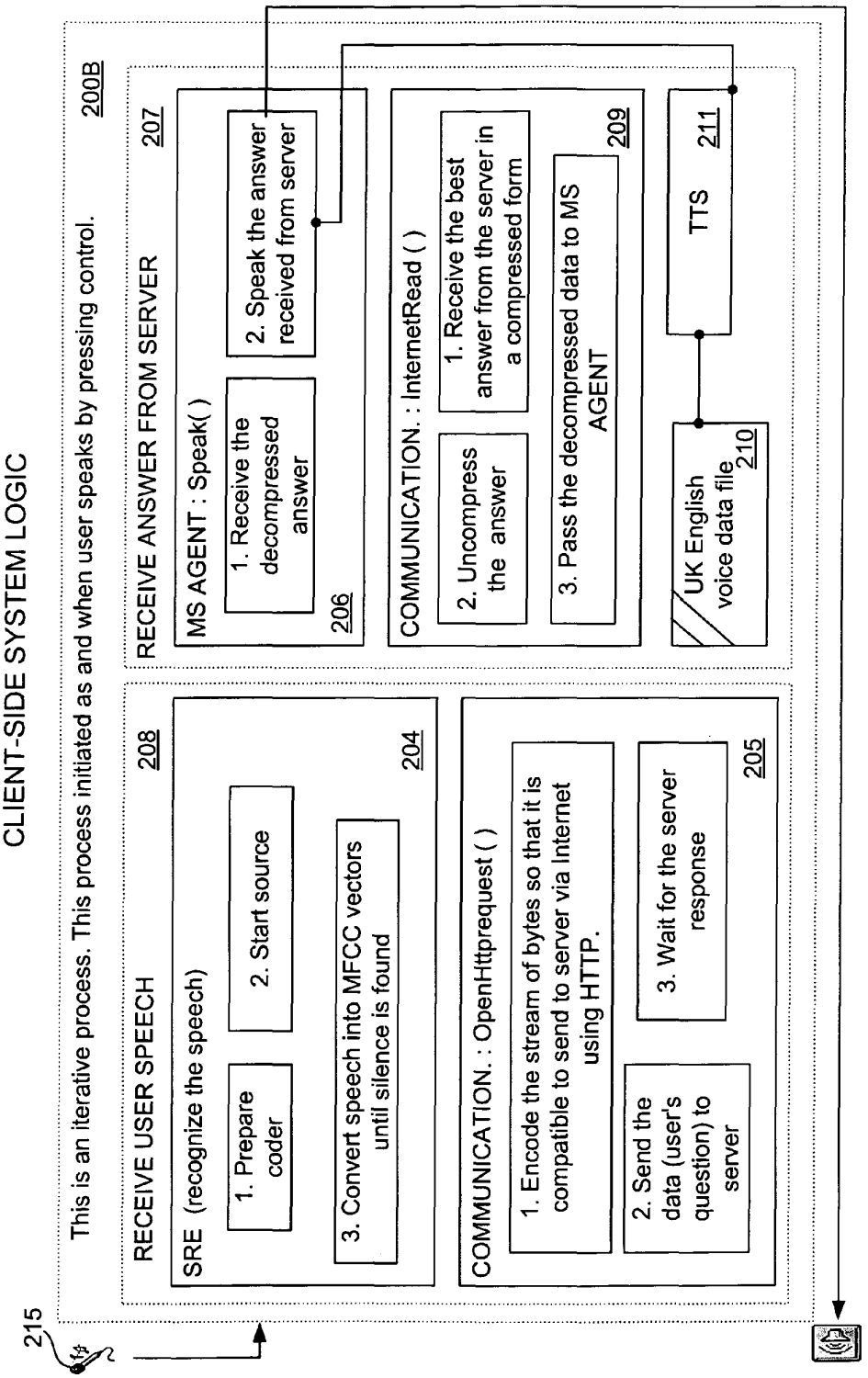


Figure 2 (Page 3/3)
CLIENT-SIDE SYSTEM LOGIC

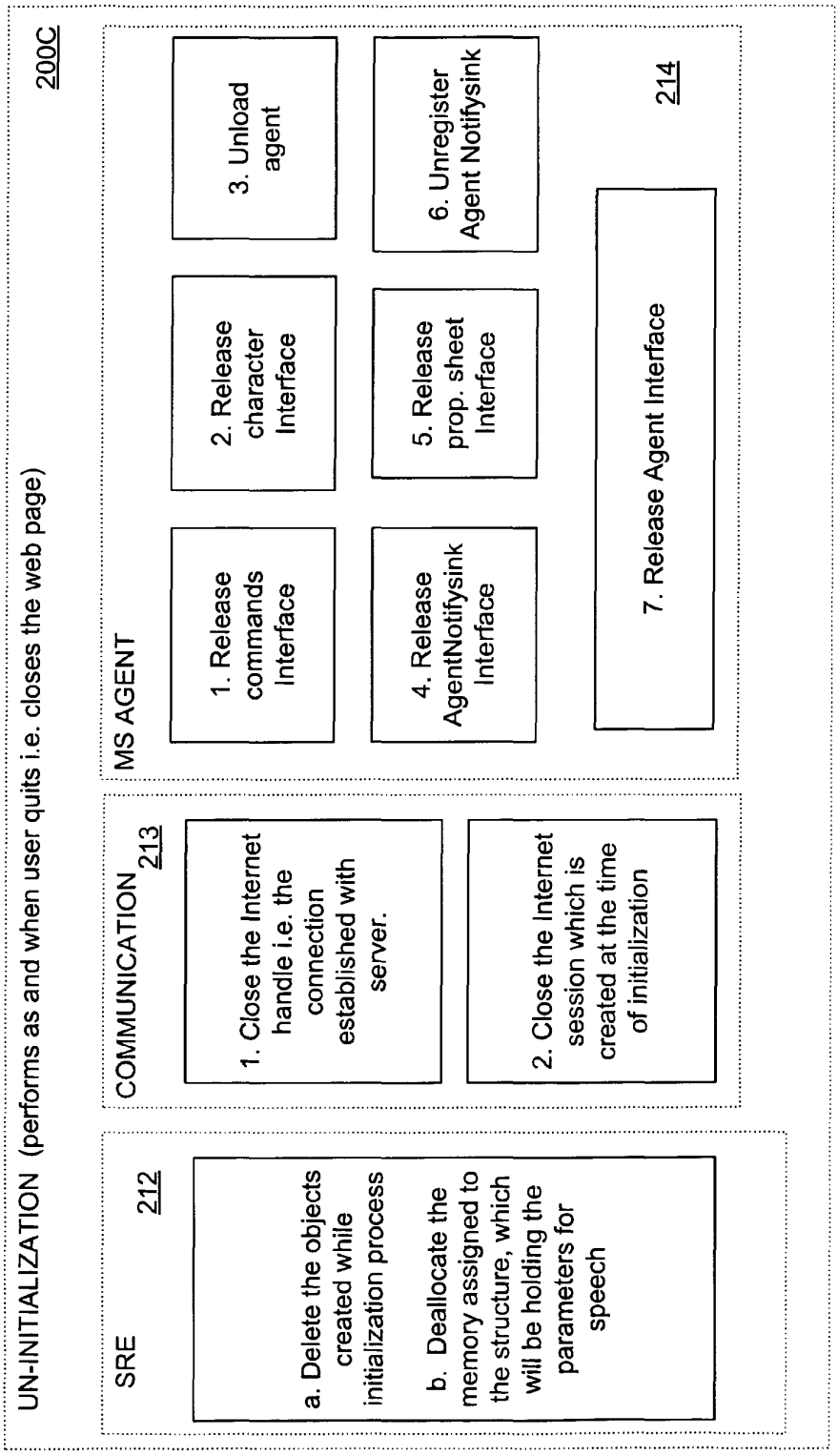


Fig. 2-2 **Client-side Initialization**

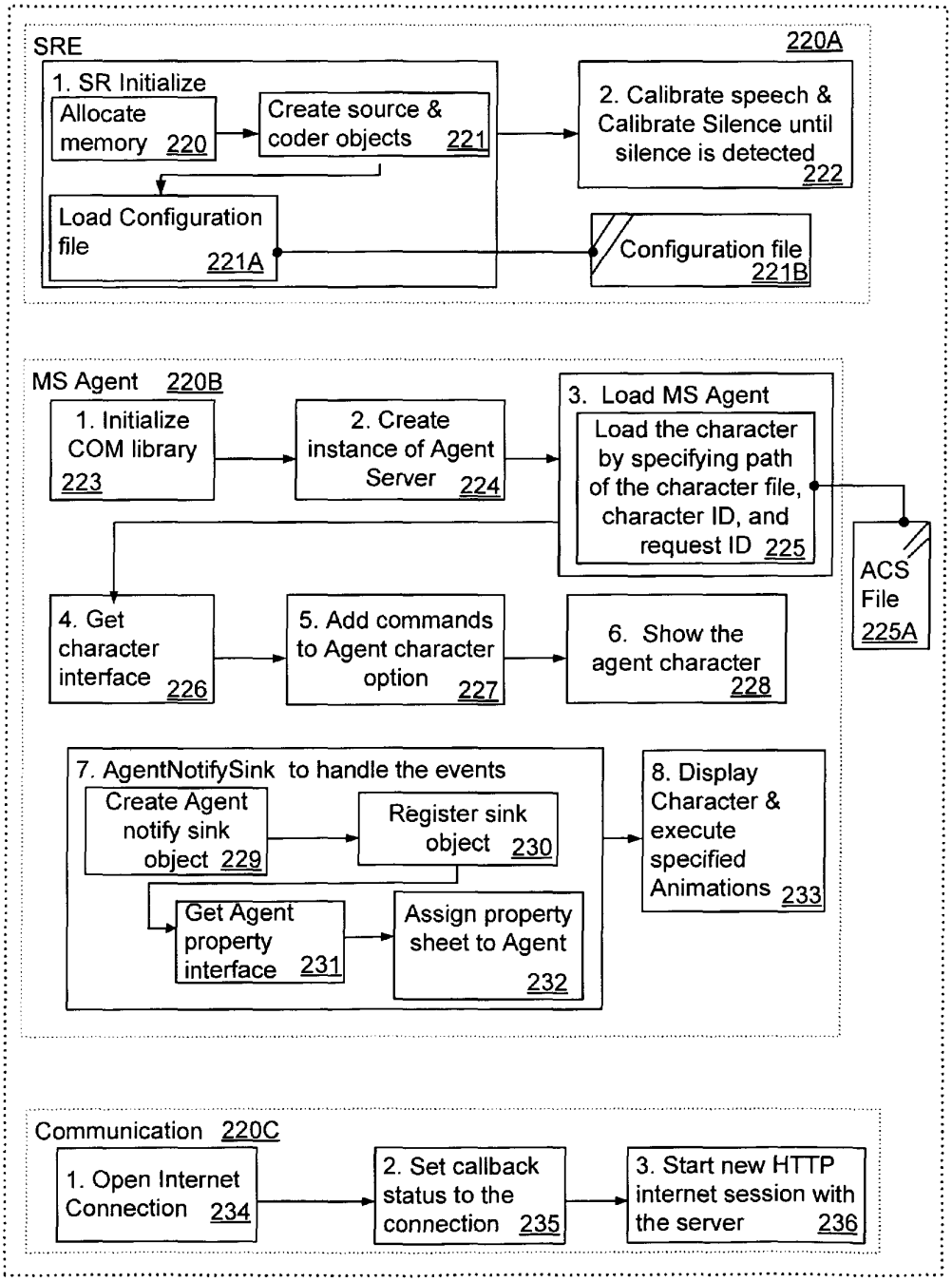


Fig. 3

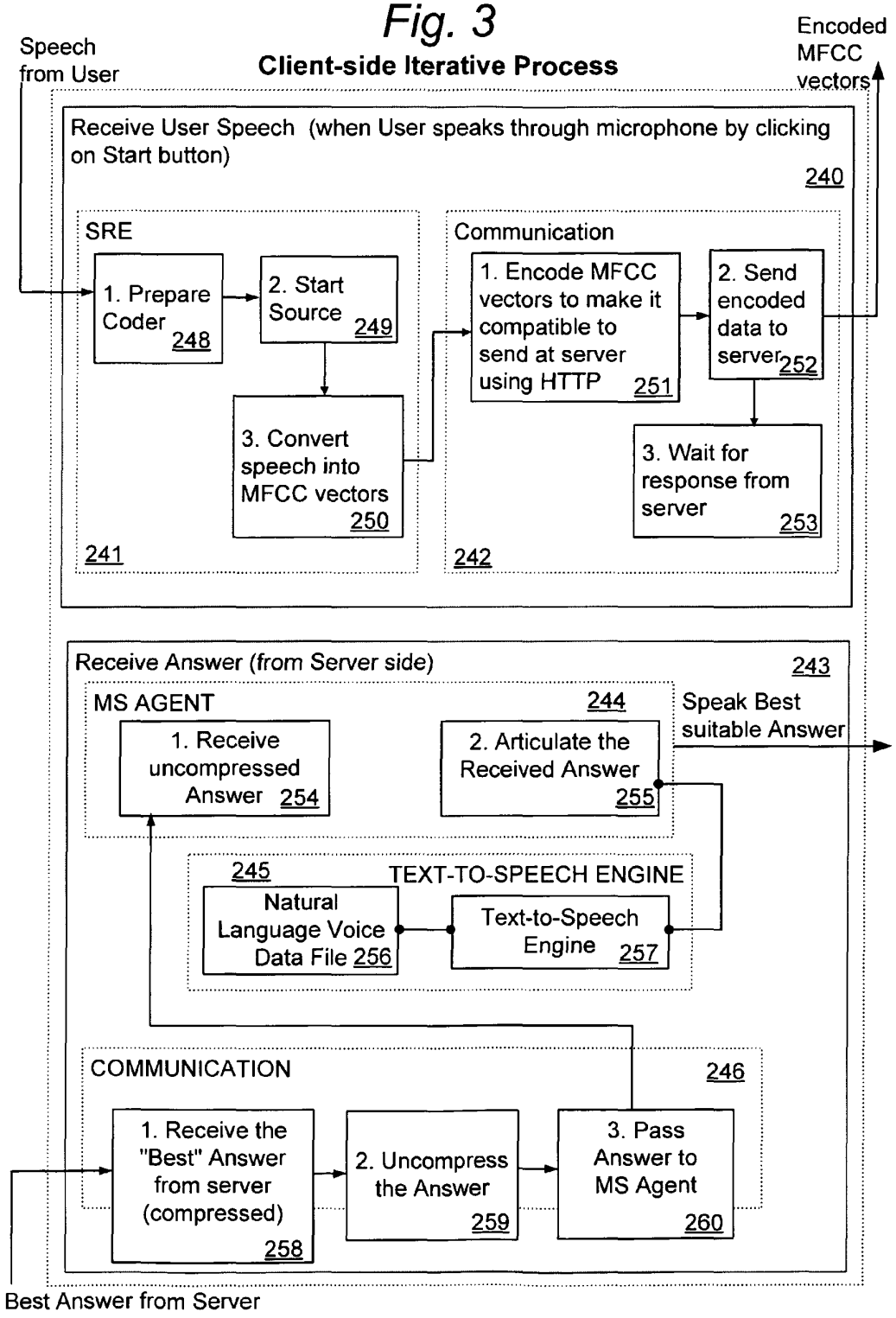

Client-side Iterative Process
Speech from User
Encoded MFCC vectors
Receive User Speech (when User speaks through microphone by clicking on Start button)
240
SRE
1. Prepare Coder 248
2. Start Source 249
3. Convert speech into MFCC vectors 250
241
Communication
1. Encode MFCC vectors to make it compatible to send at server using HTTP 251
2. Send encoded data to server 252
3. Wait for response from server 253
242
Receive Answer (from Server side)
243
MS AGENT
244
1. Receive uncompressed Answer 254
2. Articulate the Received Answer 255
Speak Best suitable Answer
245
TEXT-TO-SPEECH ENGINE
Natural Language Voice Data File 256
Text-to-Speech Engine 257
COMMUNICATION
246
1. Receive the "Best" Answer from server (compressed) 258
2. Uncompress the Answer 259
3. Pass Answer to MS Agent 260
Best Answer from Server

Fig. 4

Client-side Un-Initialization

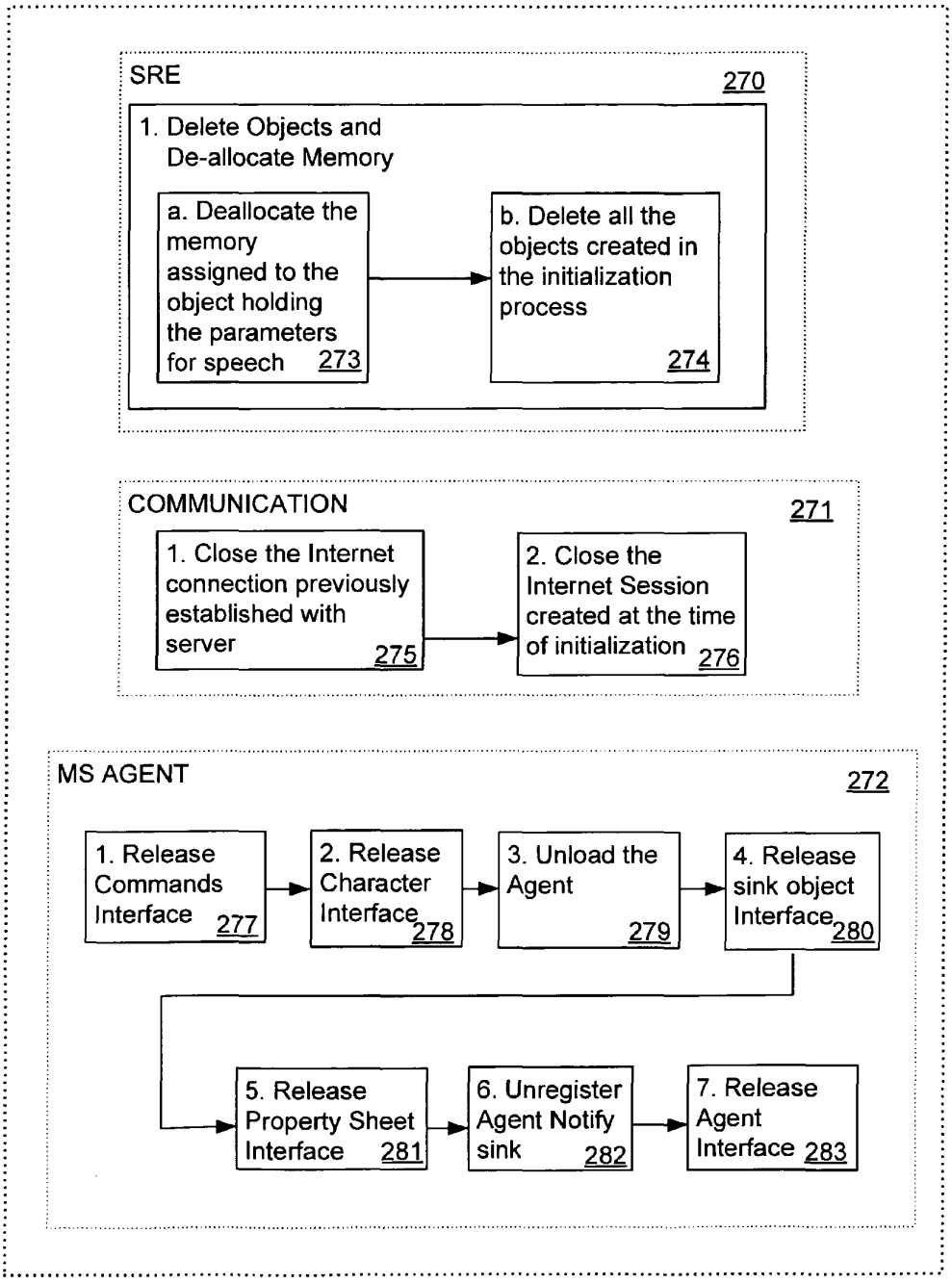


Fig. 4A

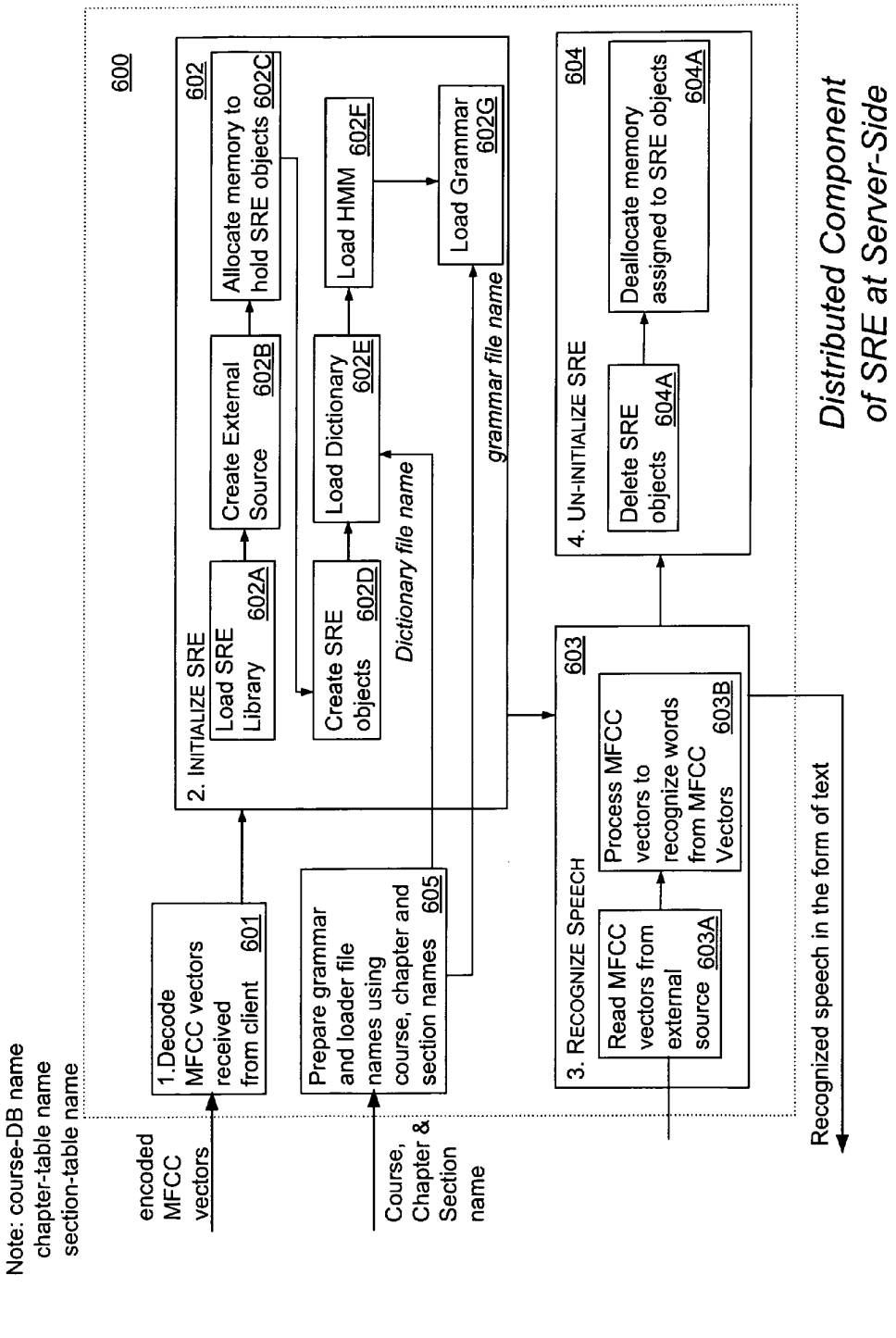

600
Distributed Component of SRE at Server-Side
Note: course-DB name
chapter-table name
section-table name
encoded MFCC vectors
Course, Chapter & Section name
Recognized speech in the form of text
1. Decode MFCC vectors received from client 601
2. INITIALIZE SRE 602
Load SRE Library 602A
Create External Source 602B
Allocate memory to hold SRE objects 602C
Create SRE objects 602D
Load Dictionary 602E
Load HMM 602F
Load Grammar 602G
Prepare grammar and loader file names using course, chapter and section names 605
Dictionary file name
grammar file name
3. RECOGNIZE SPEECH 603
Read MFCC vectors from external source 603A
Process MFCC vectors to recognize words from MFCC Vectors 603B
4. UN-INITIALIZE SRE 604
Delete SRE objects 604A
Deallocate memory assigned to SRE objects 604A

Fig. 4B

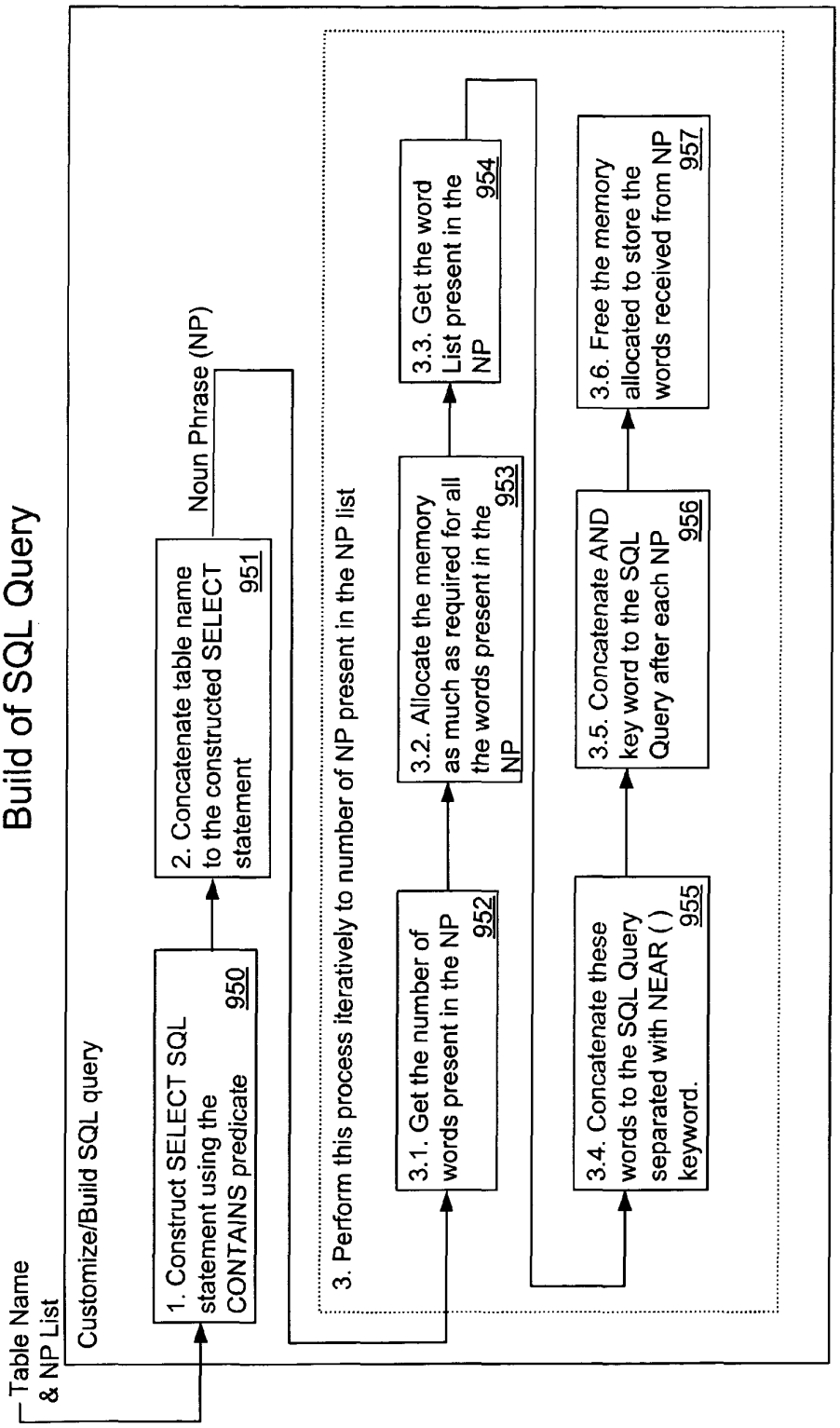

Build of SQL Query
Table Name & NP List
Customize/Build SQL query
1. Construct SELECT SQL statement using the CONTAINS predicate 950
2. Concatenate table name to the constructed SELECT statement 951
Noun Phrase (NP)
3. Perform this process iteratively to number of NP present in the NP list
3.1. Get the number of words present in the NP 952
3.2. Allocate the memory as much as required for all the words present in the NP 953
3.3. Get the word List present in the NP 954
3.4. Concatenate these words to the SQL Query separated with NEAR ( ) keyword. 955
3.5. Concatenate AND key word to the SQL Query after each NP 956
3.6. Free the memory allocated to store the words received from NP 957

Fig. 4C

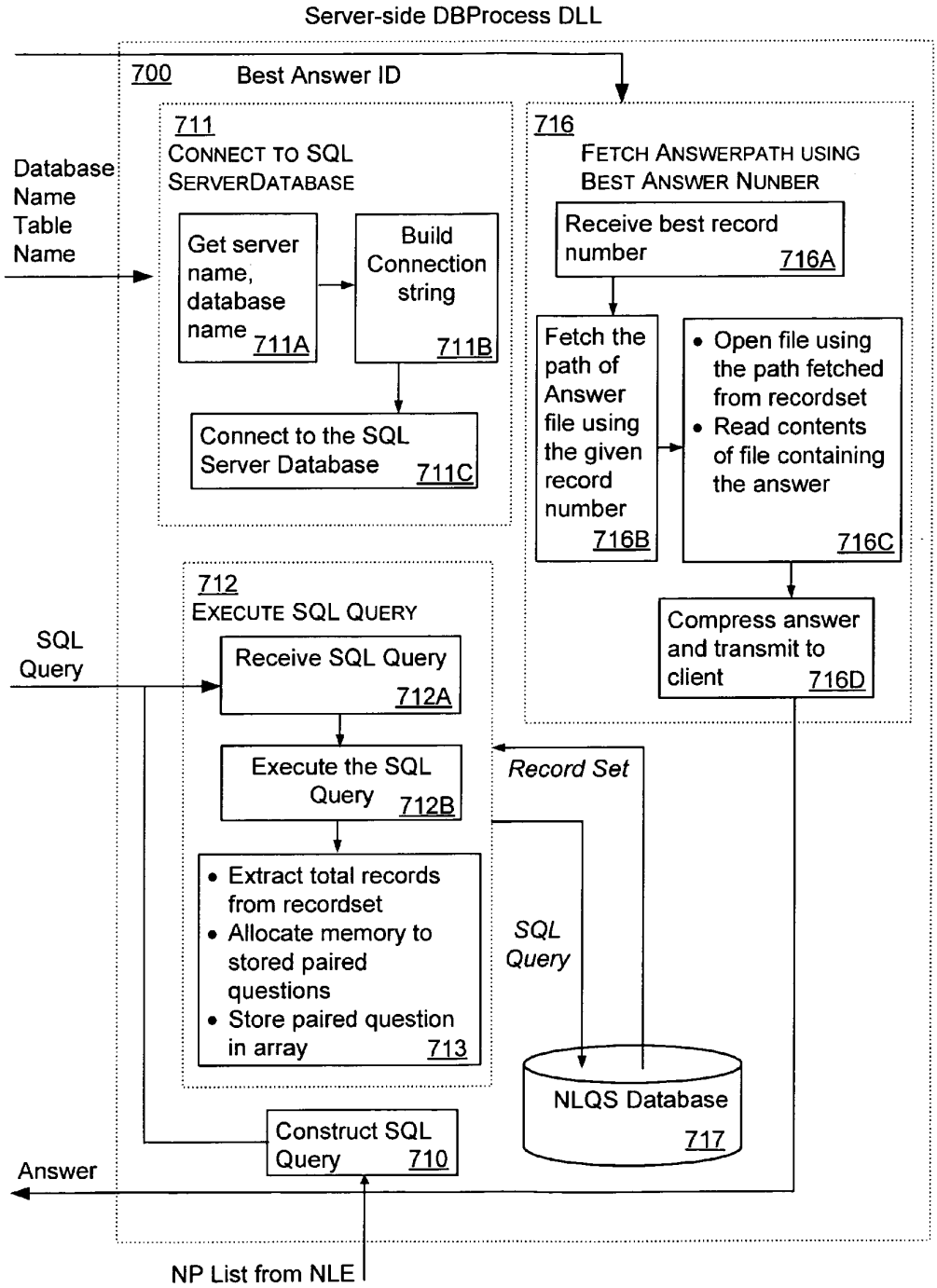

Server-side DBProcess DLL
700
Best Answer ID
711
CONNECT TO SQL SERVERDATABASE
Database Name
Table Name
Get server name, database name 711A
Build Connection string 711B
Connect to the SQL Server Database 711C
716
FETCH ANSWERPATH USING BEST ANSWER NUNBER
Receive best record number 716A
Fetch the path of Answer file using the given record number 716B
• Open file using the path fetched from recordset
• Read contents of file containing the answer 716C
Compress answer and transmit to client 716D
712
EXECUTE SQL QUERY
SQL Query
Receive SQL Query 712A
Execute the SQL Query 712B
Record Set
• Extract total records from recordset
• Allocate memory to stored paired questions
• Store paired question in array 713
SQL Query
NLQS Database 717
Construct SQL Query 710
Answer
NP List from NLE

Fig. 4D

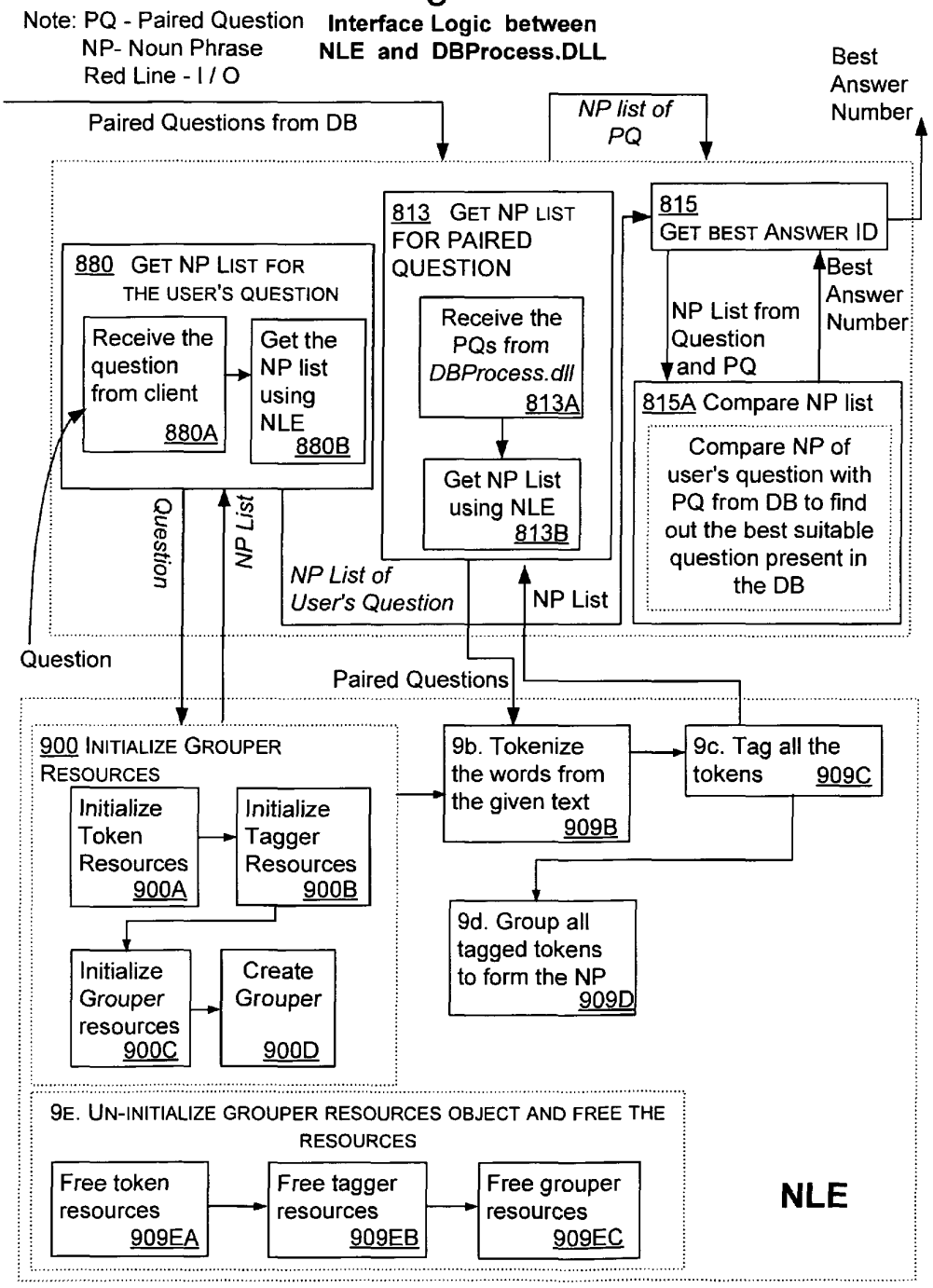

Note: PQ - Paired Question
NP- Noun Phrase
Red Line - I / O
Interface Logic between NLE and DBProcess.DLL
Paired Questions from DB
NP list of PQ
Best Answer Number
880 GET NP LIST FOR THE USER'S QUESTION
Receive the question from client 880A
Get the NP list using NLE 880B
813 GET NP LIST FOR PAIRED QUESTION
Receive the PQs from DBProcess.dll 813A
Get NP List using NLE 813B
815 GET BEST ANSWER ID
NP List from Question and PQ
Best Answer Number
815A Compare NP list
Compare NP of user's question with PQ from DB to find out the best suitable question present in the DB
Question
NP List
NP List of User's Question
NP List
Question
Paired Questions
900 INITIALIZE GROUPER RESOURCES
Initialize Token Resources 900A
Initialize Tagger Resources 900B
Initialize Grouper resources 900C
Create Grouper 900D
9b. Tokenize the words from the given text 909B
9c. Tag all the tokens 909C
9d. Group all tagged tokens to form the NP 909D
9E. UN-INITIALIZE GROUPER RESOURCES OBJECT AND FREE THE RESOURCES
Free token resources 909EA
Free tagger resources 909EB
Free grouper resources 909EC
NLE

Fig. 5

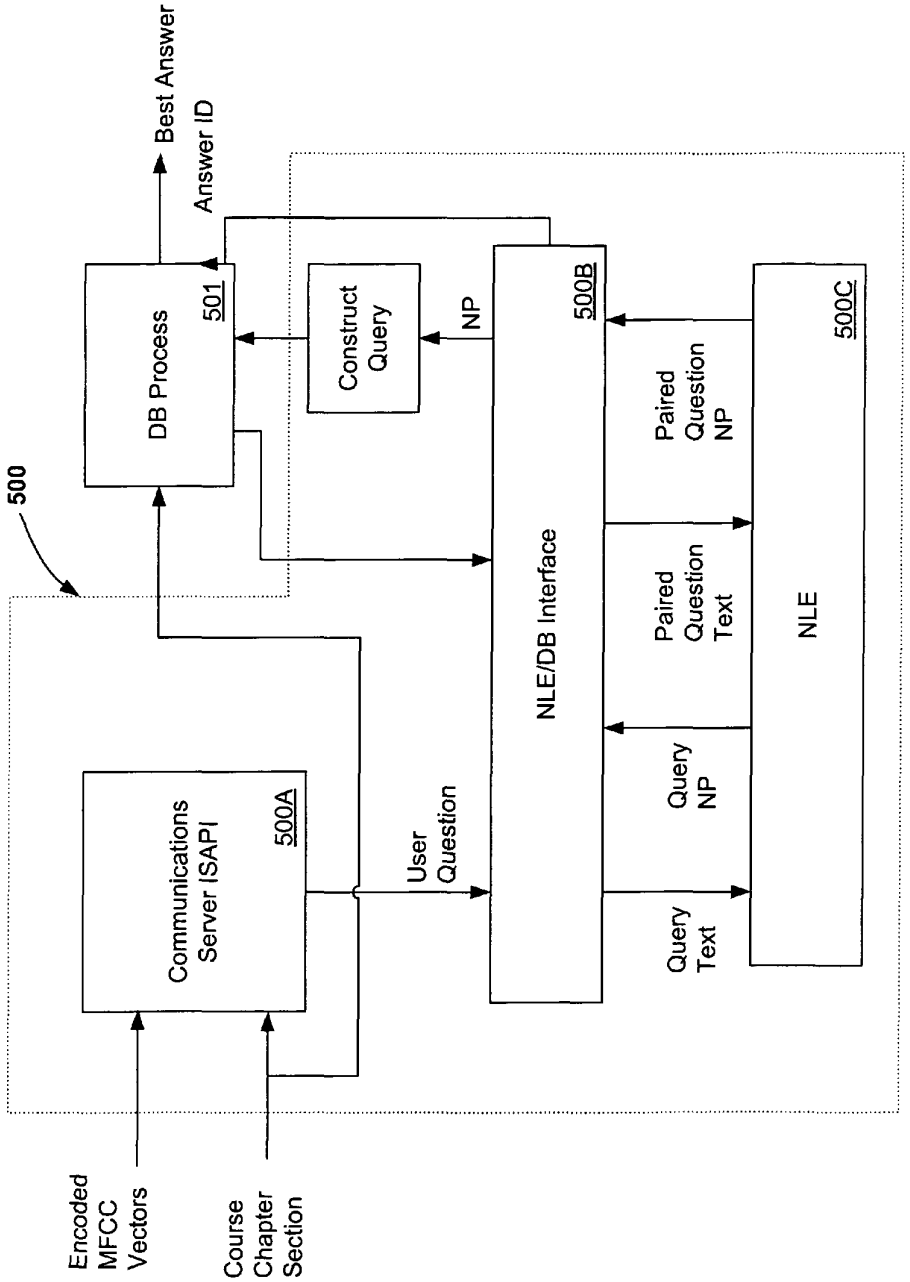

500
Best Answer
Answer ID
DB Process
501
Construct Query
NP
NLE/DB Interface
500B
Paired Question NP
Paired Question Text
Query NP
Query Text
NLE
500C
Communications Server ISAPI
500A
User Question
Encoded MFCC Vectors
Course Chapter Section

Fig.6

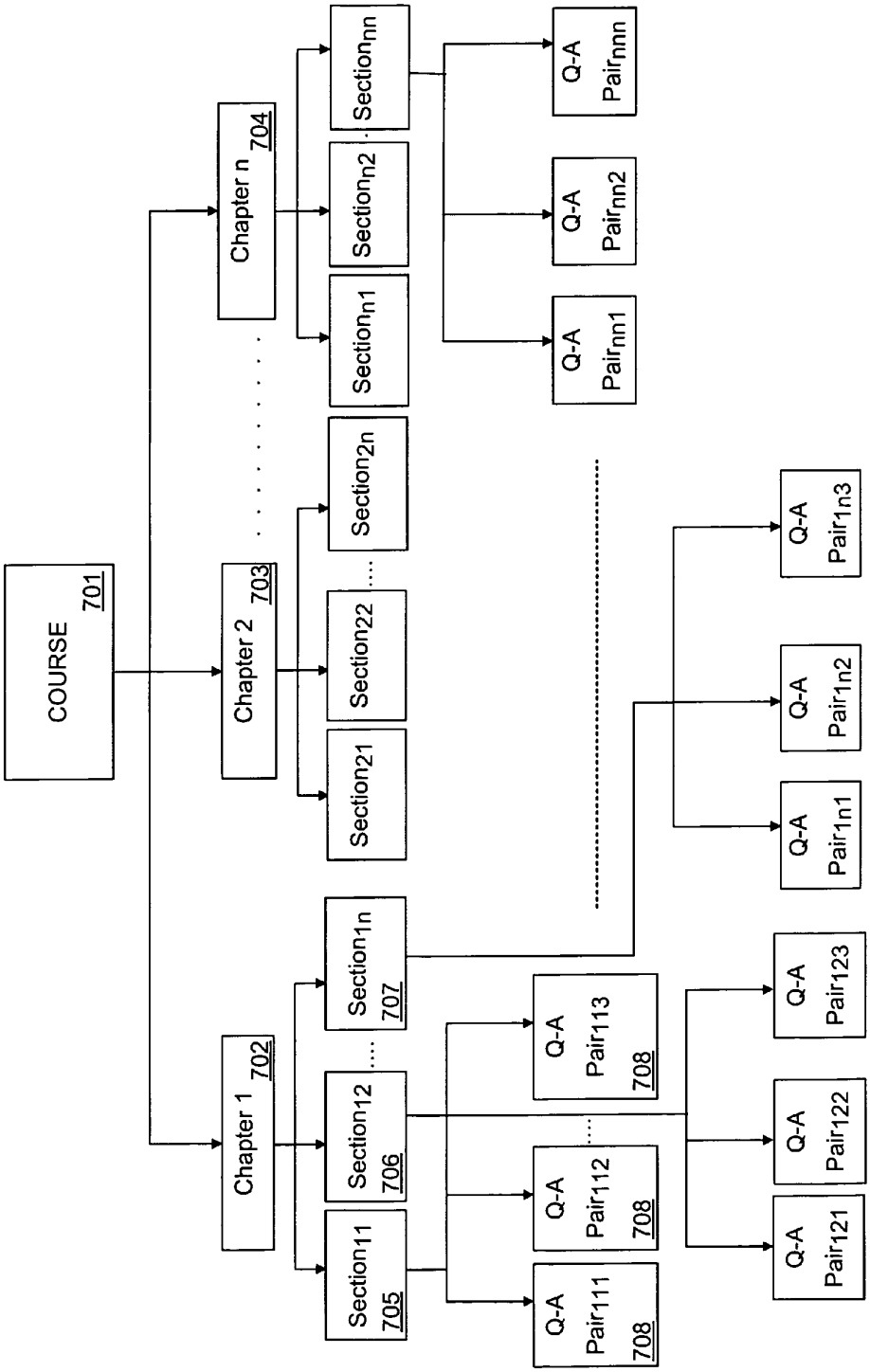

COURSE 701
Chapter 1 702
Chapter 2 703
Chapter n 704
Section11 705
Section12 706
Section1n 707
Section21
Section22
Section2n
Sectionn1
Sectionn2
Sectionnn
Q-A Pair111 708
Q-A Pair112 708
Q-A Pair113 708
Q-A Pair121
Q-A Pair122
Q-A Pair123
Q-A Pair1n1
Q-A Pair1n2
Q-A Pair1n3
Q-A Pairnn1
Q-A Pairnn2
Q-A Pairnnn

Fig. 7A

FIELD NAME 701A	DATA TYPE 702A	SIZE 703A	NULL 704A	PRIMARY KEY 705A	INDEXED? 706A
ChapterName 707A	Varchar	255	No	No	Yes
SectionName 708A	Varchar	255	No	No	Yes

Fig. 7B

FIELD NAME 720	DATA TYPE 721	SIZE 722	NULL 723	PRIMARY KEY 724	INDEXED? 725
Chapter_ID 726	Integer		No	Yes	Yes
Answer_ID 727	Char	5	No	UNIQUE	Yes
Section_Name 728	Varchar	255	No	UNIQUE	Yes
Answer_Title 729	Varchar	255	Yes	No	Yes
PairedQuestion 730	Text	16	No	No	Yes (Full-Text)
AnswerPath 731	Varchar	255	No	No	Yes
Creator 732	Varchar	50	No	No	Yes
Date_of_Creation 733	Date	-	No	No	Yes
Date_of_Modification 734	Date	-	No	No	Yes

Fig. 7C

Field 720	Description 735
AnswerID 727	An integer - automatically incremented for user convenience
Section_Name 728	Name of section to which the particular record belongs. This field along with AnswerID has to be made primary key
Answer_Title 729	A short description of the answer
PairedQuestion 730	Contains one or more combinations of questions for the related answer whose path is stored in the next column AnswerPath
AnswerPath 731	Contains the path of text file, which contains theanswer to the rleated qusetions stored in the previous column
Creator 732	Name of content creator
Date_of_Creation 733	Date on which content has been added
Date_of_Modification 734	Date on which content has been changed or modified

Fig. 7D

FIELD 740	DATA TYPE 741	SIZE 742	NULL 743	PRIMARY KEY 744	INDEXED 745
Answer_ID 746	Char	5	No	Yes	Yes
Answer_Title 747	Varchar	255	Yes	No	No
PairedQuestion 748	Text	16	No	No	Yes (Full-Text)
Answer_Path 749	Varchar	255	No	No	No
Creator 750	Varchar	50	No	No	No
Date_of_Creation 751	Date	-	No	No	No
Date_of_Modification 752	Date	-	No	No	No

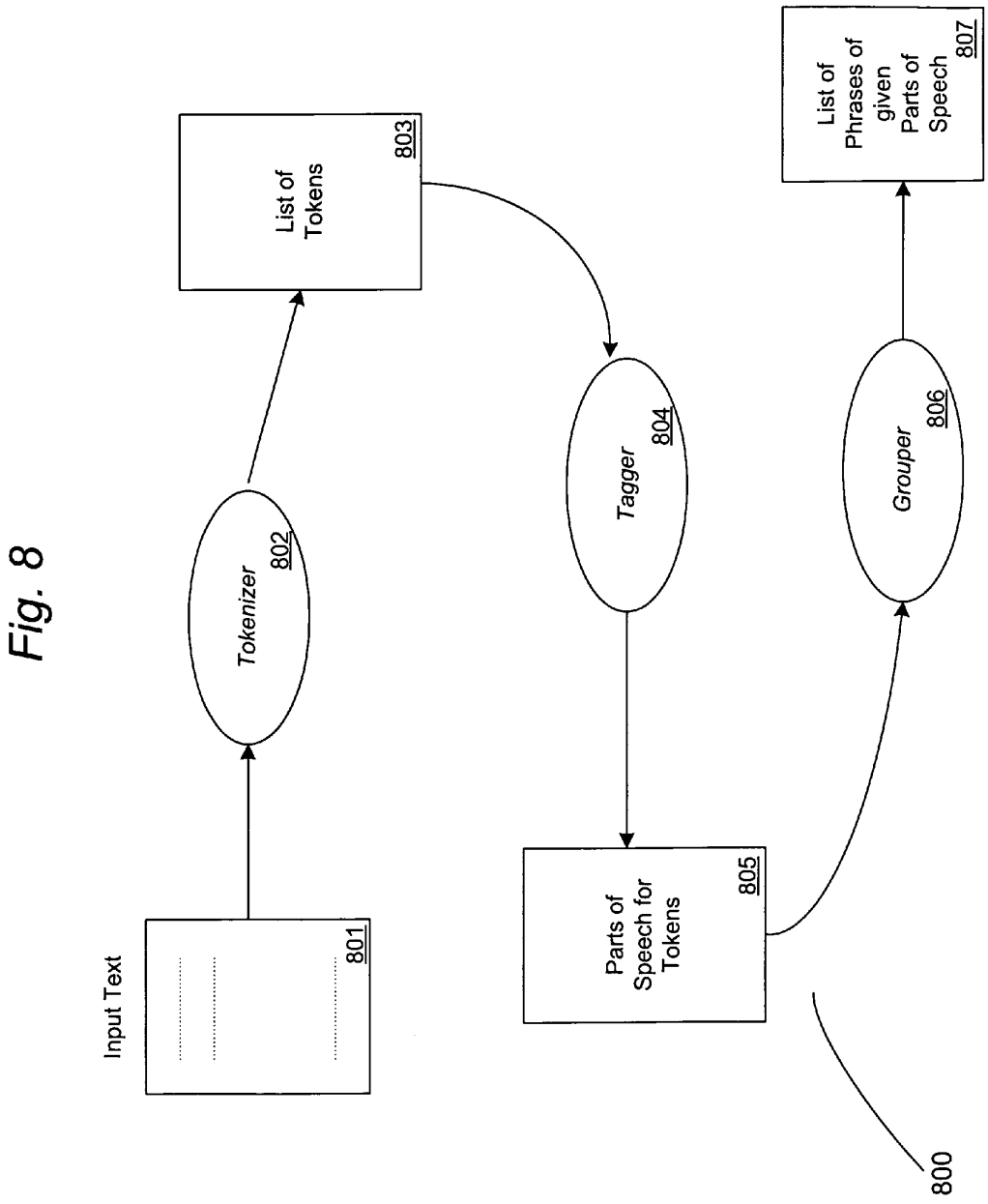

Input Text
801
Tokenizer
802
List of Tokens
803
Tagger
804
Parts of Speech for Tokens
805
Grouper
806
List of Phrases of given Parts of Speech
807
800

Fig. 8

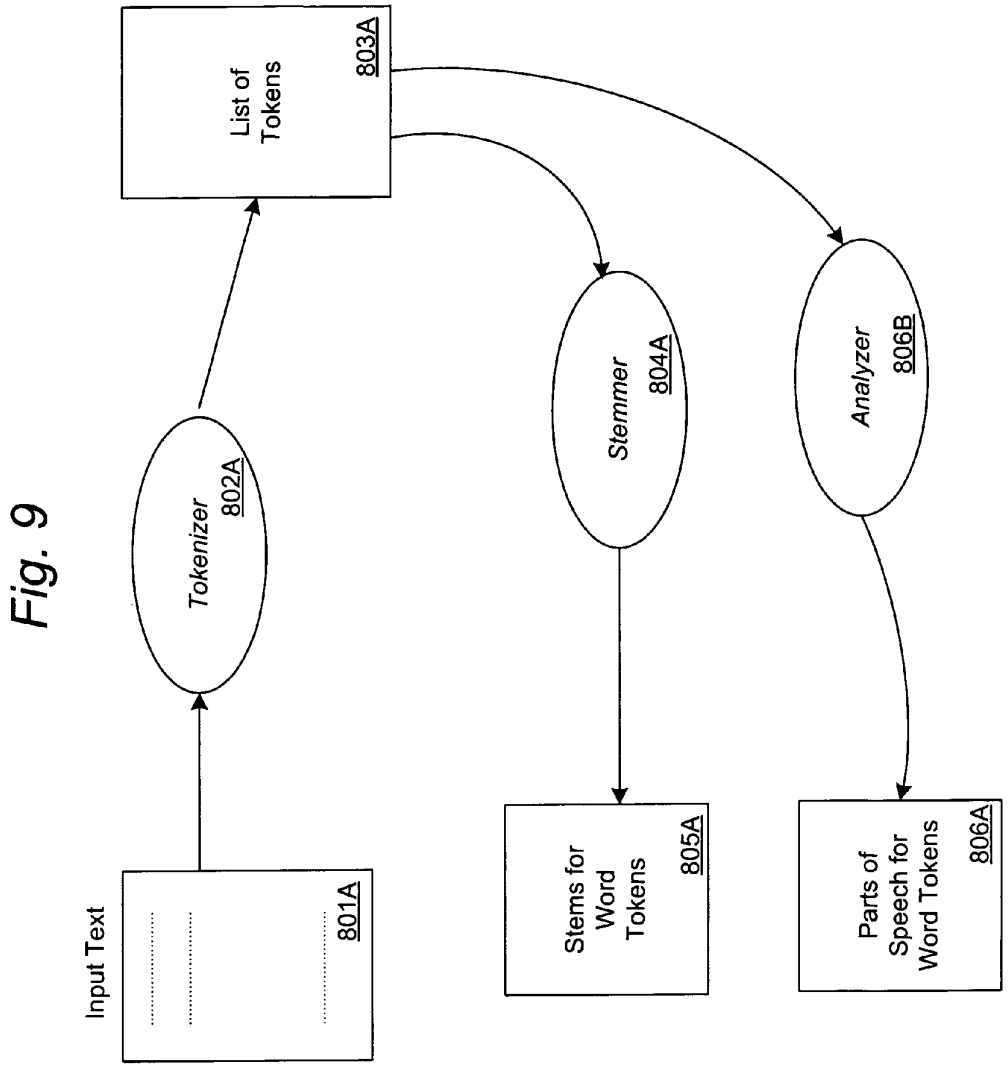

Input Text
801A
Tokenizer
802A
List of Tokens
803A
Stemmer
804A
Stems for Word Tokens
805A
Analyzer
806B
Parts of Speech for Word Tokens
806A

Fig. 9

Fig. 10

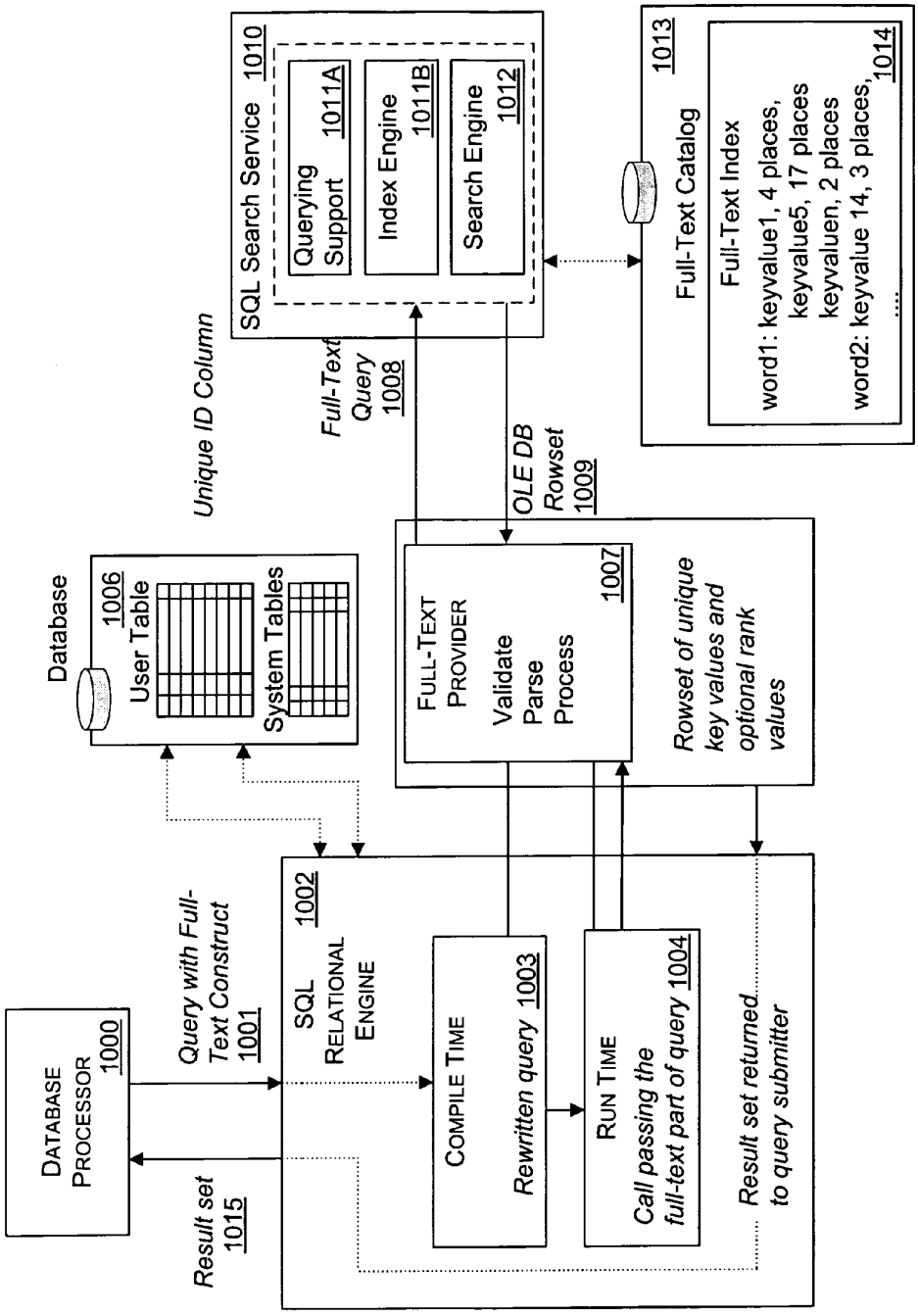

DATABASE PROCESSOR 1000
Query with Full-Text Construct 1001
Result set 1015
SQL RELATIONAL ENGINE 1002
COMPILE TIME
Rewritten query 1003
RUN TIME
Call passing the full-text part of query 1004
Result set returned to query submitter
FULL-TEXT PROVIDER 1007
Validate
Parse
Process
Rowset of unique key values and optional rank values
OLE DB Rowset 1009
Full-Text Query 1008
Unique ID Column
Database
User Table 1006
System Tables
SQL Search Service 1010
Querying Support 1011A
Index Engine 1011B
Search Engine 1012
Full-Text Catalog 1013
Full-Text Index
word1: keyvalue1, 4 places,
keyvalue5, 17 places
keyvaluen, 2 places
word2: keyvalue 14, 3 places, 1014

Fig. 11A

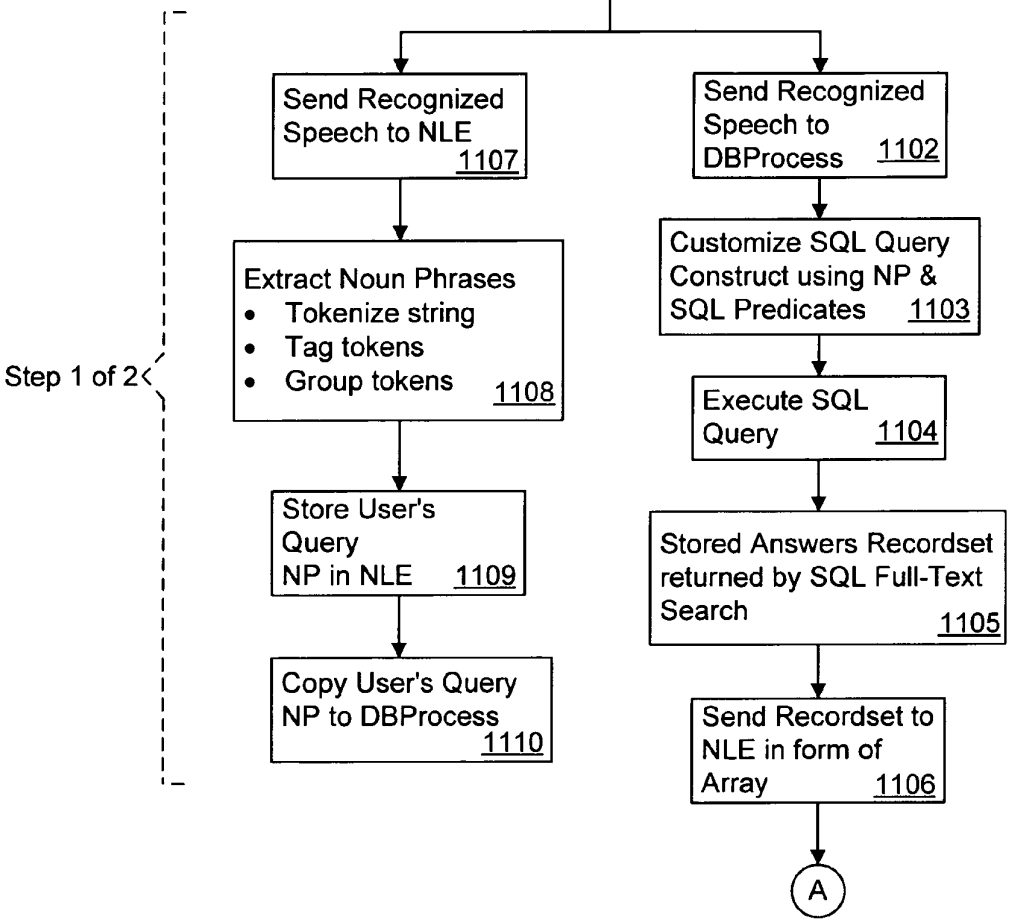

Speech Recognition of User's Query 1101
Send Recognized Speech to NLE 1107
Send Recognized Speech to DBProcess 1102
Extract Noun Phrases
• Tokenize string
• Tag tokens
• Group tokens 1108
Customize SQL Query Construct using NP & SQL Predicates 1103
Execute SQL Query 1104
Store User's Query NP in NLE 1109
Stored Answers Recordset returned by SQL Full-Text Search 1105
Copy User's Query NP to DBProcess 1110
Send Recordset to NLE in form of Array 1106
A
Step 1 of 2

Fig. 11B

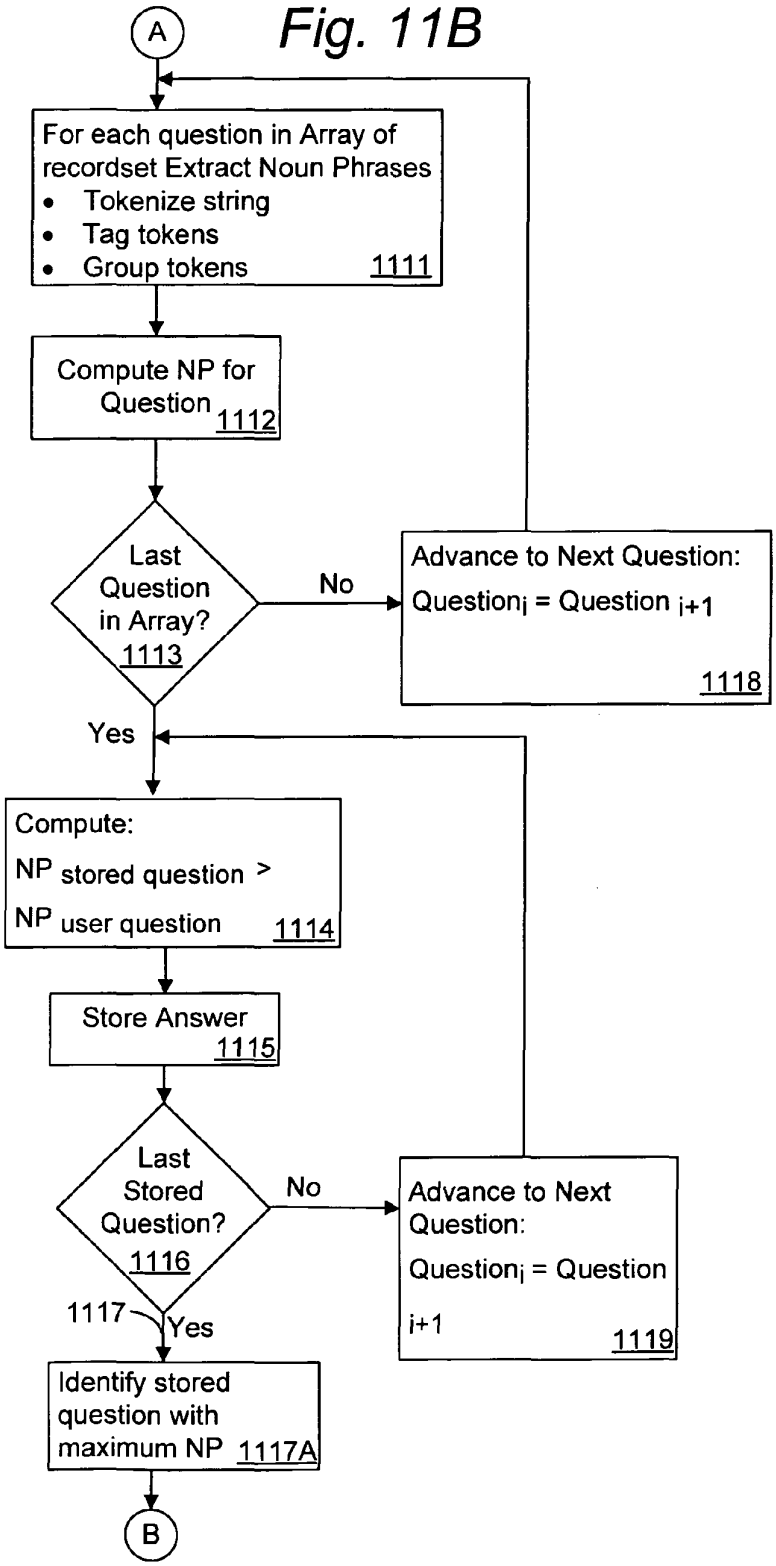

A
For each question in Array of recordset Extract Noun Phrases
• Tokenize string
• Tag tokens
• Group tokens
1111
Compute NP for Question
1112
Last Question in Array?
1113
No
Advance to Next Question:
Question$_i$ = Question $_{i+1}$
1118
Yes
Compute:
NP stored question >
NP user question
1114
Store Answer
1115
Last Stored Question?
1116
No
Advance to Next Question:
Question$_i$ = Question $_{i+1}$
1119
1117
Yes
Identify stored question with maximum NP
1117A
B

Fig. 11C

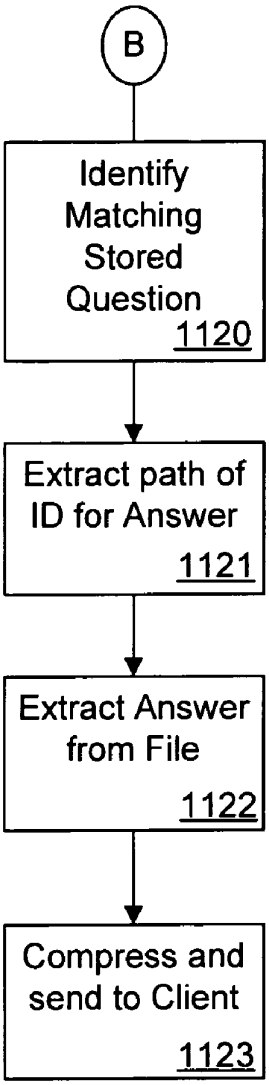

B
Identify Matching Stored Question
1120
Extract path of ID for Answer
1121
Extract Answer from File
1122
Compress and send to Client
1123

Fig. 12

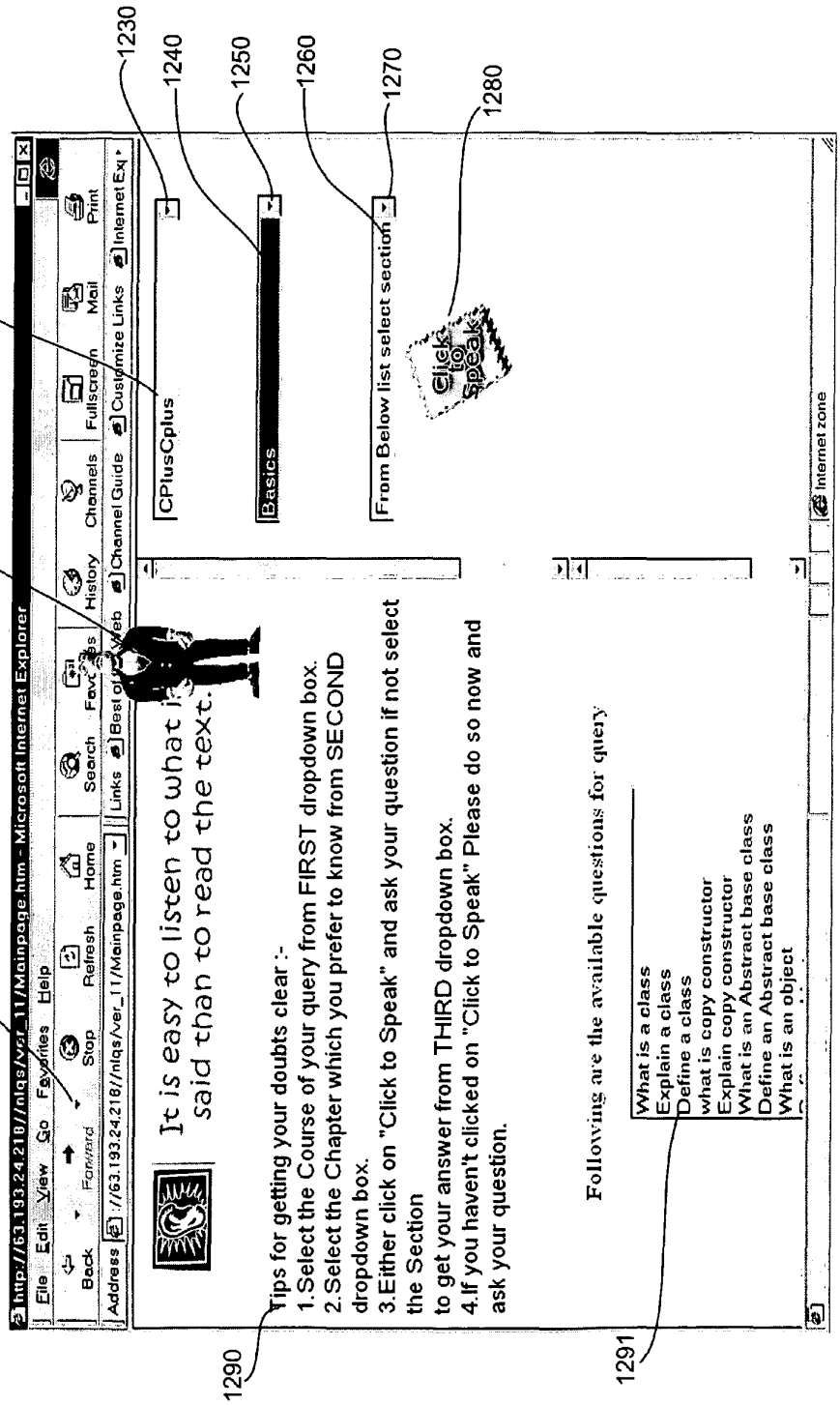

1200
1210
1220
1230
1240
1250
1260
1270
1280
1290
1291
CPlusCplus
Basics
From Below list select section
Click to Speak
It is easy to listen to what is said than to read the text.
Tips for getting your doubts clear :-
1.Select the Course of your query from FIRST dropdown box.
2.Select the Chapter which you prefer to know from SECOND dropdown box.
3.Either click on "Click to Speak" and ask your question if not select the Section
to get your answer from THIRD dropdown box.
4.If you haven't clicked on "Click to Speak" Please do so now and ask your question.
Following are the available questions for query
What is a class
Explain a class
Define a class
what is copy constructor
Explain copy constructor
What is an Abstract base class
Define an Abstract base class
What is an object
Internet zone

Fig. 13

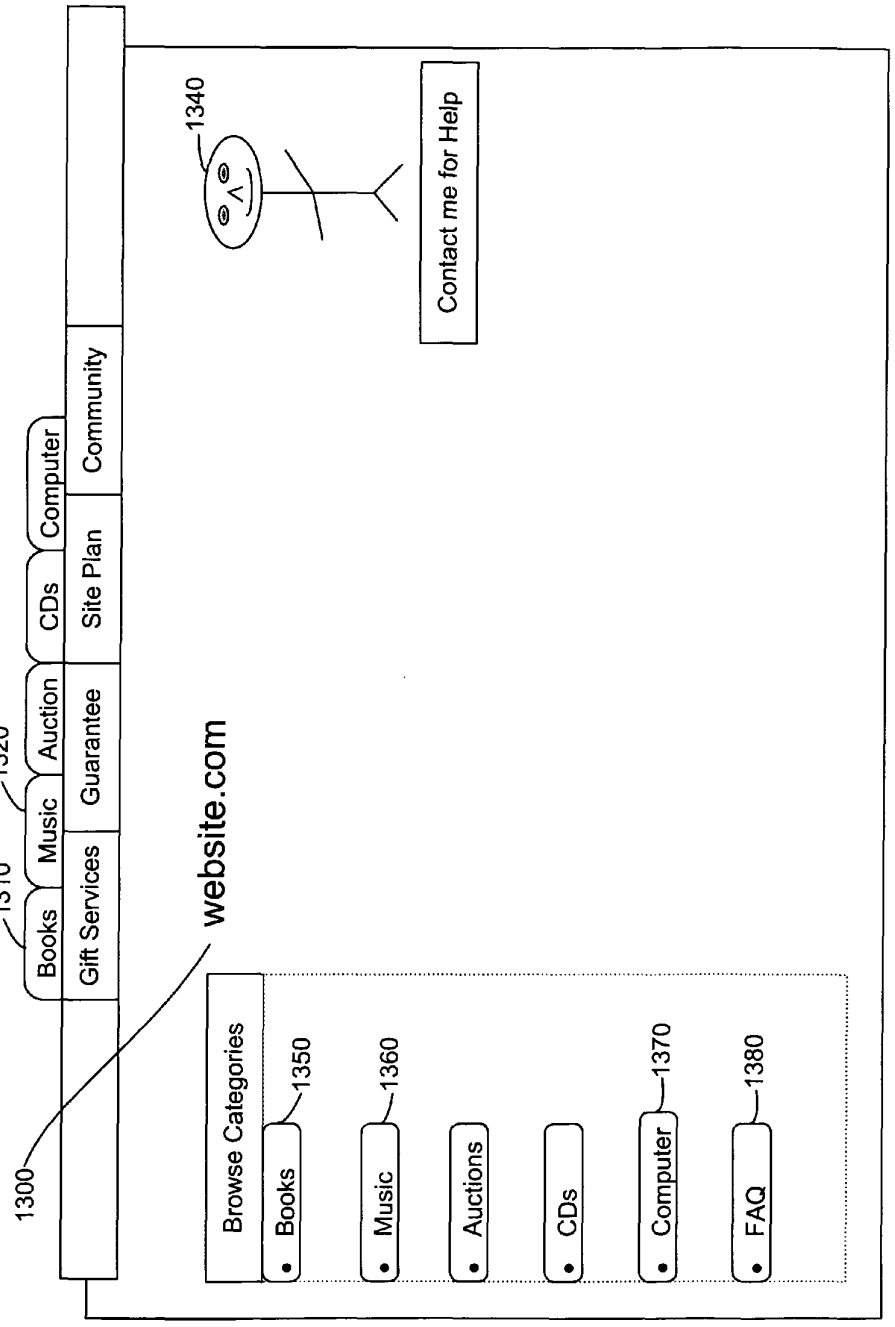

1330
Account Info
Help
1320
1310
Books
Music
Auction
CDs
Computer
Gift Services
Guarantee
Site Plan
Community
website.com
1300
1340
Contact me for Help
Browse Categories
Books
1350
Music
1360
Auctions
CDs
Computer
1370
FAQ
1380

Fig. 14

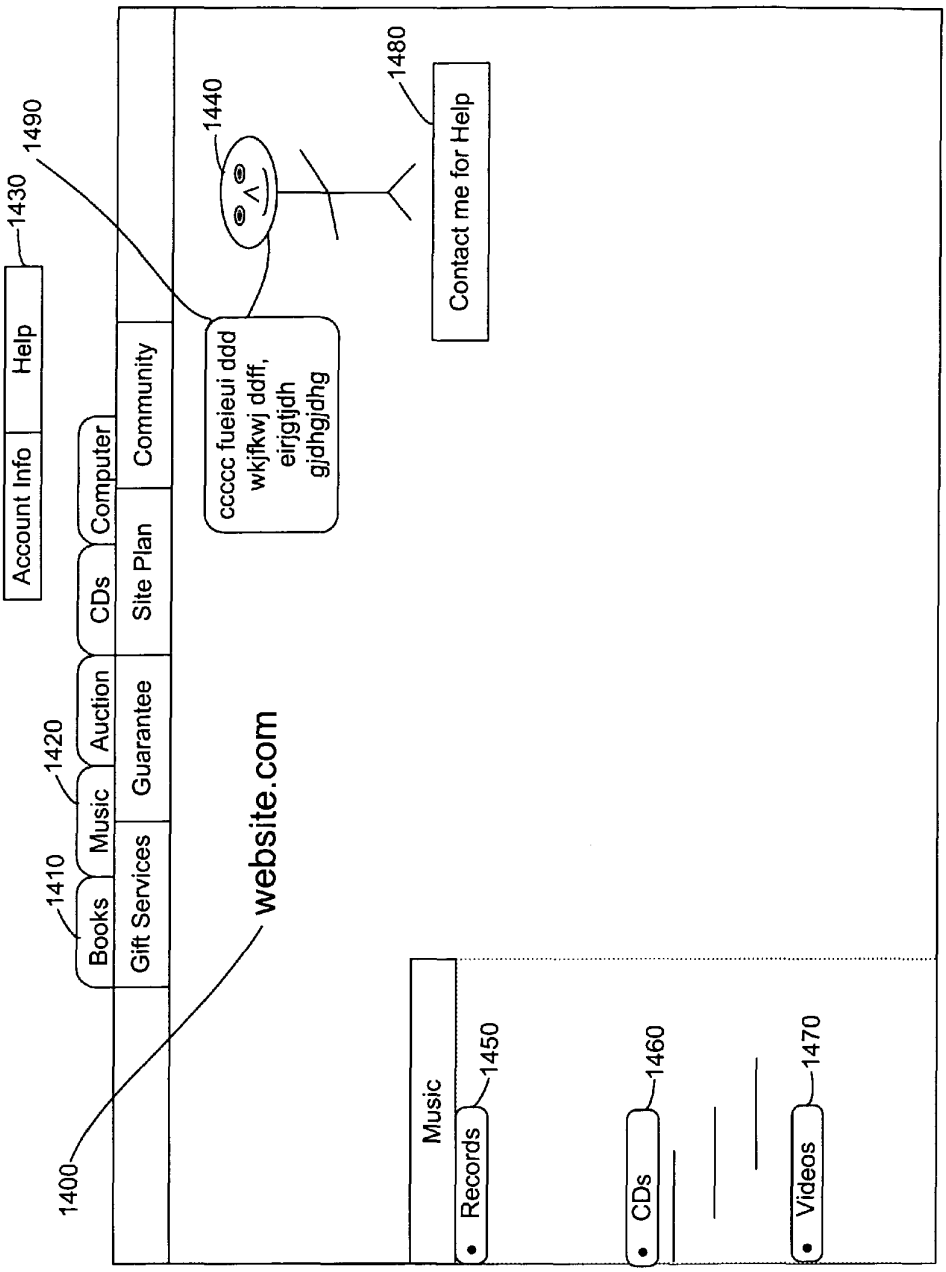

1400
website.com
1410
Books
Music
Gift Services
1420
Auction
Guarantee
CDs
Site Plan
Computer
Community
Account Info
Help
1430
1490
cccc fueieui ddd wkjfkwj ddff, eirjgtjdh gjdhgjdhg
1440
1480
Contact me for Help
Music
Records
1450
CDs
1460
Videos
1470

Fig. 15

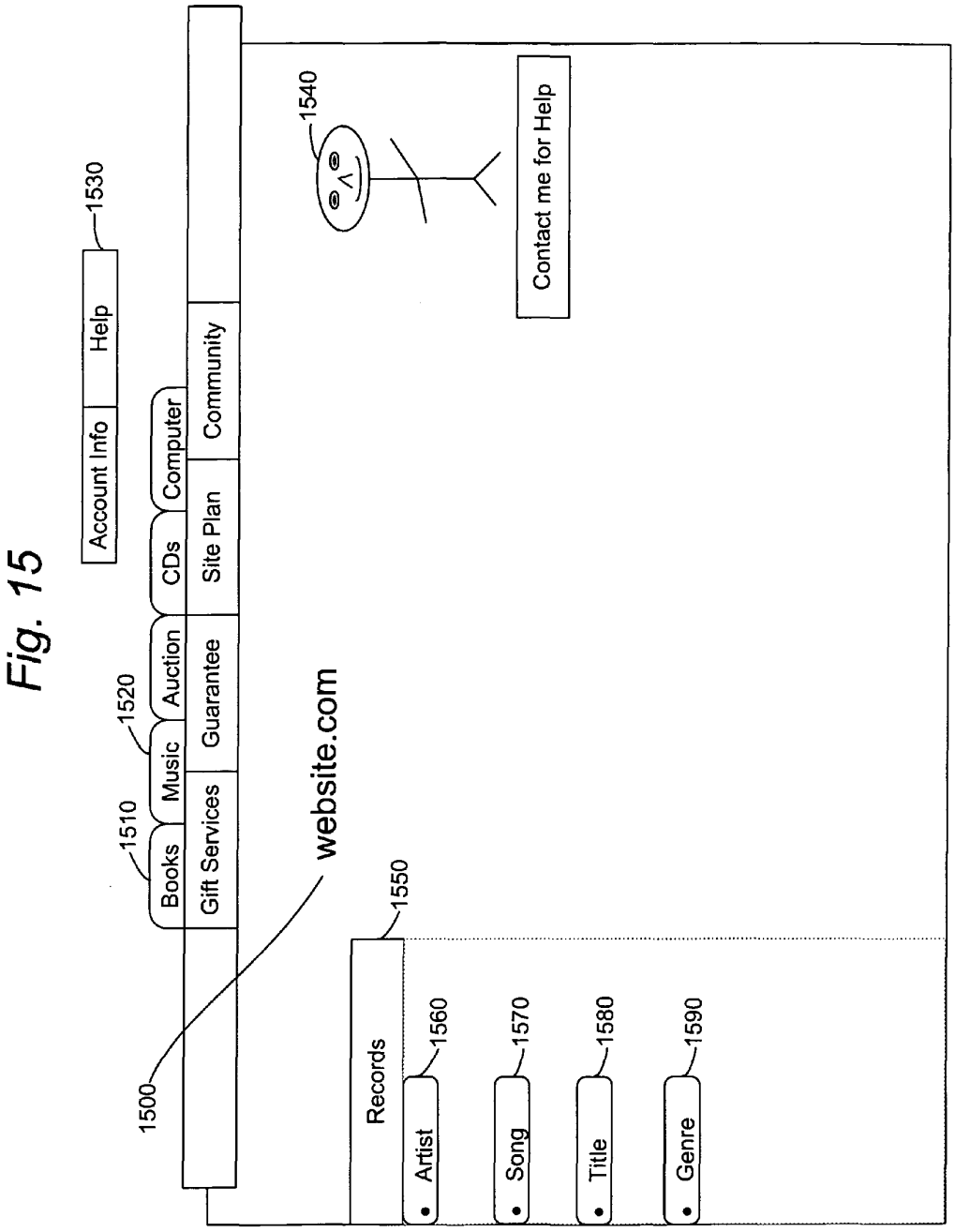

1500
website.com
1510
Books
Music
1520
Auction
Guarantee
Gift Services
CDs
Computer
Site Plan
Community
Account Info
Help
1530
1540
Contact me for Help
1550
Records
Artist
1560
Song
1570
Title
1580
Genre
1590

Fig. 16

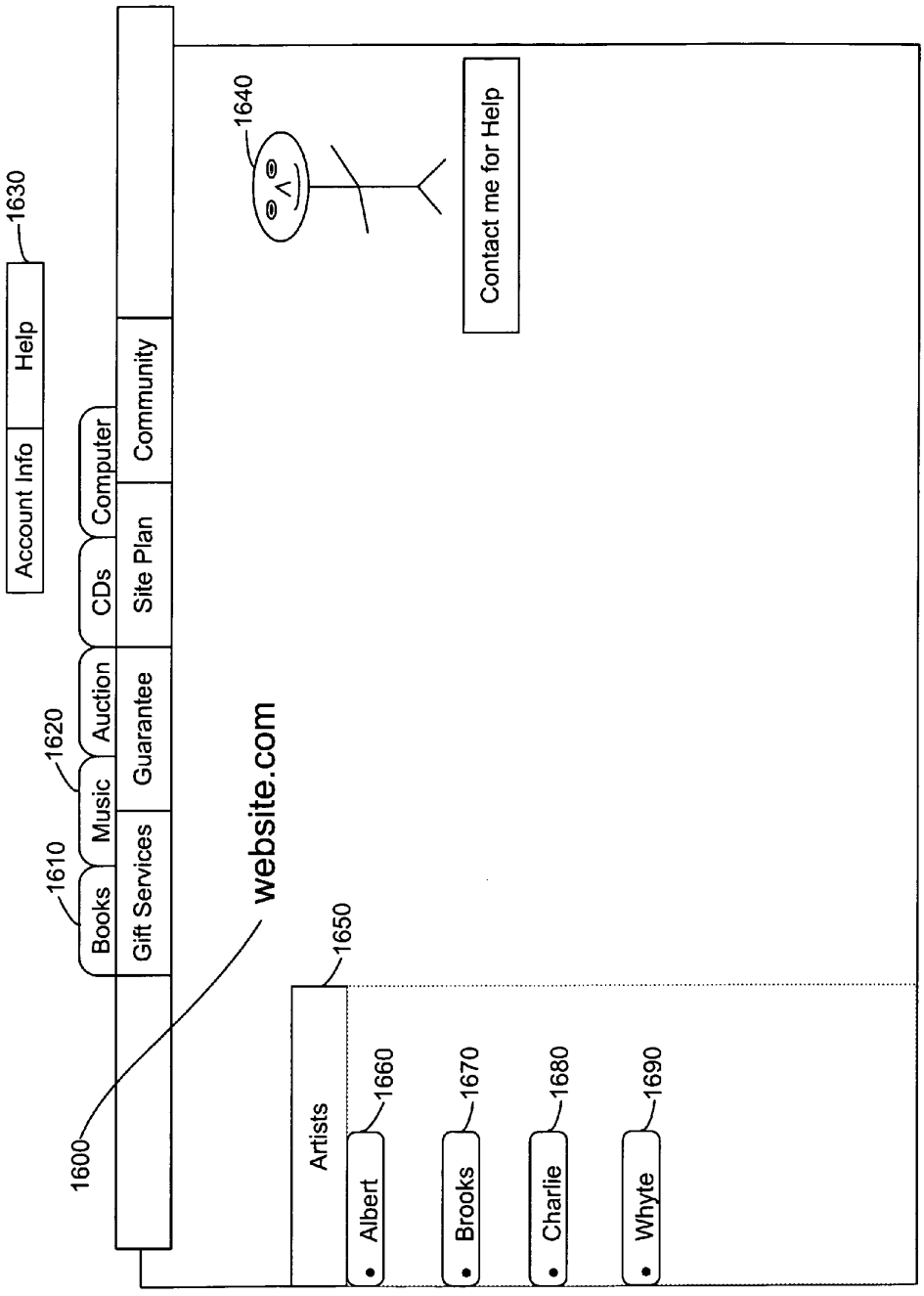

1600
website.com
1610
Books
Music
Gift Services
1620
Auction
Guarantee
CDs
Computer
Site Plan
Community
Account Info
Help
1630
1640
Contact me for Help
1650
Artists
1660
Albert
1670
Brooks
1680
Charlie
1690
Whyte

Fig. 17

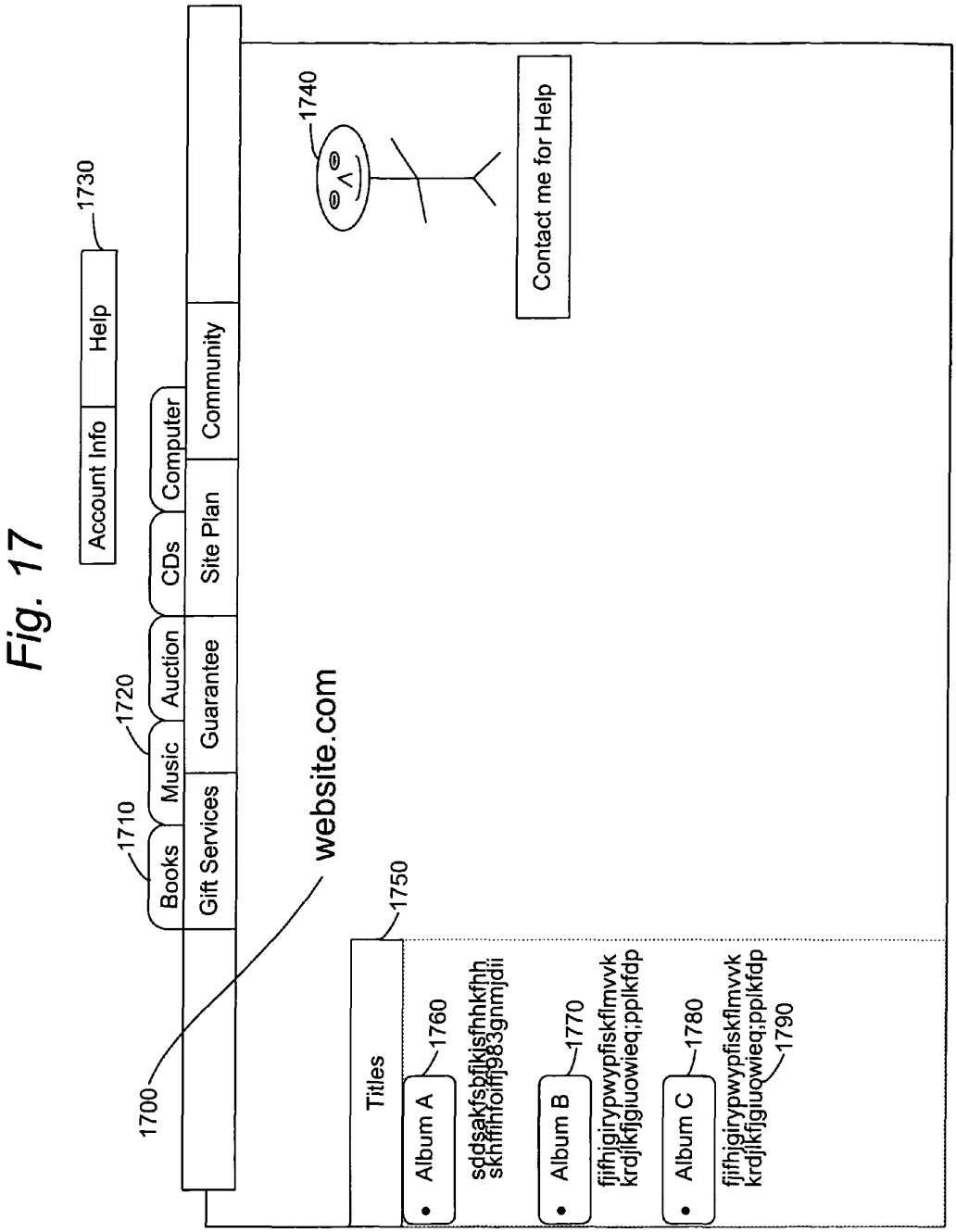

1700
website.com
1710
Books
Music
Gift Services
1720
Auction
Guarantee
CDs
Site Plan
Computer
Community
Account Info
Help
1730
1740
Contact me for Help
1750
Titles
Album A
1760
sddsakfsbfikjsfhhkfhh
skhfffhfoiffj983gnmjdii
Album B
1770
fjifhjgirypwypfiskflmvvk
krdjlkfjgiuowieq;pplkfdp
Album C
1780
fjifhjgirypwypfiskflmvvk
krdjlkfjgiuowieq;pplkfdp
1790

Fig. 18 (Page 1/2)

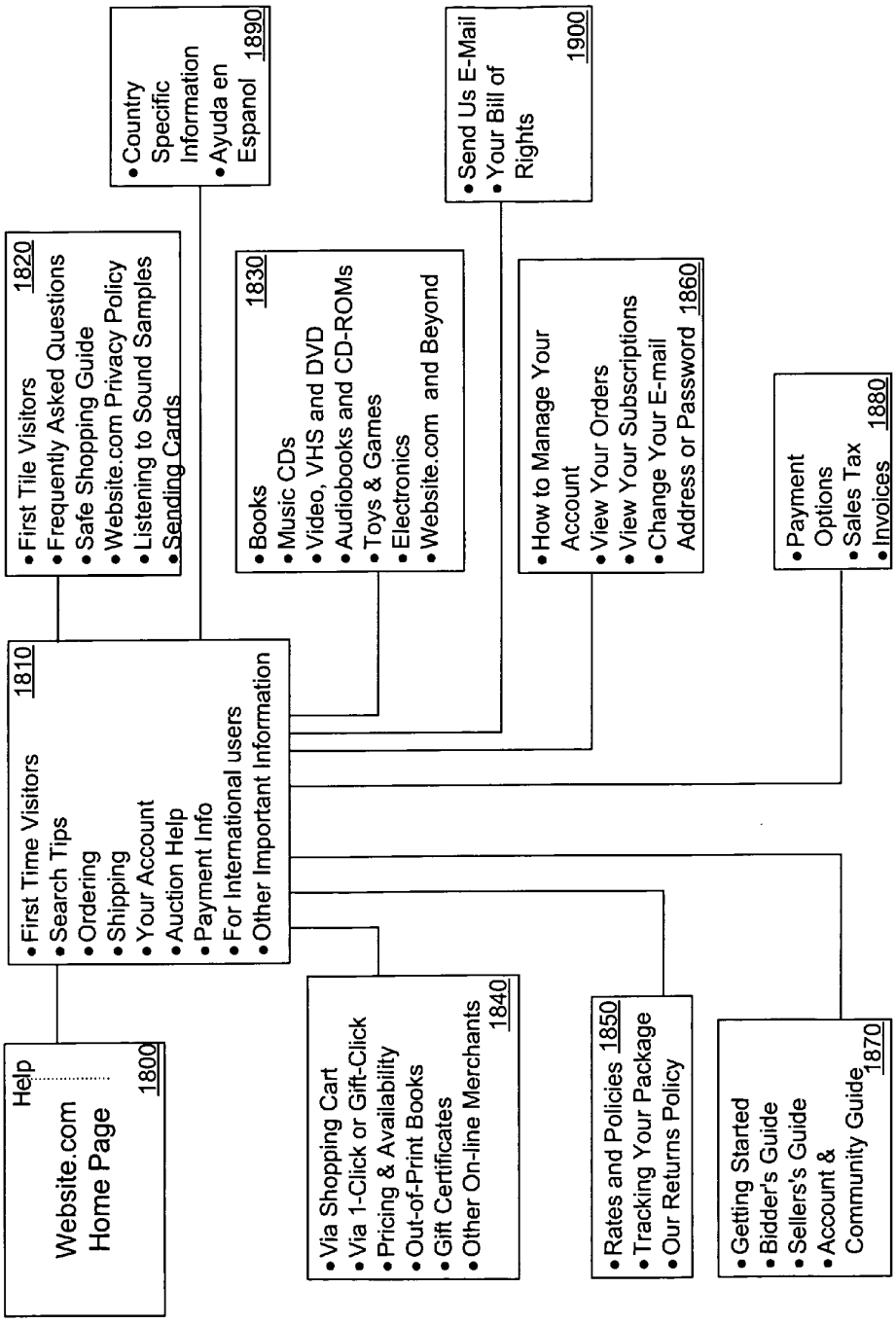
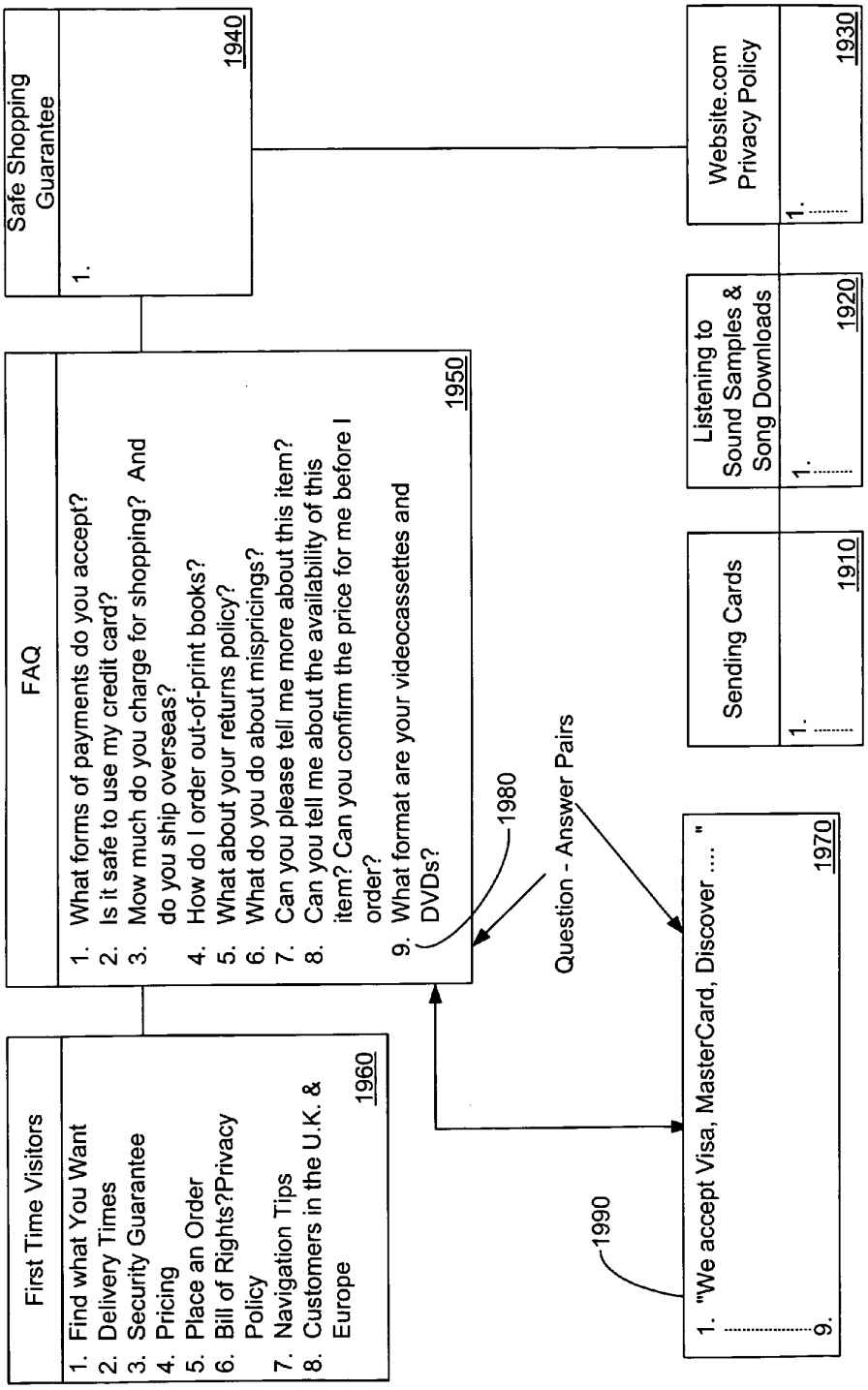


Fig. 18 (Page 2/2)



SPEECH-ENABLED SERVER FOR INTERNET WEBSITE AND METHOD

RELATED APPLICATIONS

The present application is related to the following applications also filed contemporaneously herewith:

1) Ser. No. 09/439,145 entitled Distributed Real Time Speech Recognition System;
2) Ser. No. 09/439,173 entitled Speech Based Learning/Training System;
3) Ser. No. 09/439,060 entitled Intelligent Query Engine For Processing Voice Based Queries;

The above are incorporated by reference herein.

FIELD OF THE INVENTION

The invention relates to a system and an interactive method for an enabling a website to have interactive, real-time speech-enabled web pages. This interactive system is especially useful when implemented for e-commerce, e-support, search engines and the like, so that a user can intelligently and easily control an internet session using a conventional browser that is enhanced to handle speech capabilities.

BACKGROUND OF THE INVENTION

The INTERNET, and in particular, the World-Wide Web (WWW), is growing in popularity and usage for both commercial and recreational purposes, and this trend is expected to continue. This phenomenon is being driven, in part, by the increasing and widespread use of personal computer systems and the availability of low cost INTERNET access.

The emergence of inexpensive INTERNET access devices and high speed access techniques such as ADSL, cable modems, satellite modems, and the like, are expected to further accelerate the mass usage of the WWW.

Accordingly, it is expected that the number of entities offering services, products, etc., over the WWW will increase dramatically over the coming years. Until now, however, the INTERNET "experience" for users has been limited mostly to non-voice based input/output devices, such as keyboards, intelligent electronic pads, mice, trackballs, printers, monitors, etc. This presents somewhat of a bottleneck for interacting over the WWW for a variety of reasons.

First, there is the issue of familiarity. Many kinds of applications lend themselves much more naturally and fluently to a voice-based environment. For instance, most people shopping for audio recordings are very comfortable with asking a live sales clerk in a record store for information on titles by a particular author, where they can be found in the store, etc. While it is often possible to browse and search on one's own to locate items of interest, it is usually easier and more efficient to get some form of human assistance first, and, with few exceptions, this request for assistance is presented in the form of a oral query. In addition, many persons cannot or will not, because of physical or psychological barriers, use any of the aforementioned conventional I/O devices. For example, many older persons cannot easily read the text presented on WWW pages, or understand the layout/hierarchy of menus, or manipulate a mouse to make finely coordinated movements to indicate their selections. Many others are intimidated by the look and complexity of computer systems, WWW pages, etc., and therefore do not attempt to use online services for this reason as well.

Thus, applications which can mimic normal human interactions are likely to be preferred by potential on-line shoppers and persons looking for information over the WWW. It is also expected that the use of voice-based systems will increase the universe of persons willing to engage in e-commerce, e-learning, etc. To date, however, there are very few systems, if any, which permit this type of interaction, and, if they do, it is very limited. For example, various commercial programs sold by IBM (VIAVOICE™) and Kurzweil (DRAGON™) permit some user control of the interface (opening, closing files) and searching (by using previously trained URLs) but they do not present a flexible solution that can be used by a number of users across multiple cultures and without time consuming voice training. Typical prior efforts to implement voice based functionality in an INTERNET context can be seen in U.S. Pat. No. 5,819,220 incorporated by reference herein.

Another issue presented by the lack of voice-based systems is efficiency. Many companies are now offering technical support over the INTERNET, and some even offer live operator assistance for such queries. While this is very advantageous (for the reasons mentioned above) it is also extremely costly and inefficient, because a real person must be employed to handle such queries. This presents a practical limit that results in long wait times for responses or high labor overheads. An example of this approach can be seen U.S. Pat. No. 5,802,526 also incorporated by reference herein. In general, a service presented over the WWW is far more desirable if it is "scalable," or, in other words, able to handle an increasing amount of user traffic with little if any perceived delay or troubles by a prospective user.

In a similar context, while remote learning has become an increasingly popular option for many students, it is practically impossible for an instructor to be able to field questions from more than one person at a time. Even then, such interaction usually takes place for only a limited period of time because of other instructor time constraints. To date, however, there is no practical way for students to continue a human-like question and answer type dialog after the learning session is over, or without the presence of the instructor to personally address such queries.

Conversely, another aspect of emulating a human-like dialog involves the use of oral feedback. In other words, many persons prefer to receive answers and information in audible form. While a form of this functionality is used by some websites to communicate information to visitors, it is not performed in a real-time, interactive question-answer dialog fashion so its effectiveness and usefulness is limited.

Yet another area that could benefit from speech-based interaction involves so-called "search" engines used by INTERNET users to locate information of interest at web sites, such as the those available at YAHOO®.com, METACRAWLER®.com, EXCITE®.com, etc. These tools permit the user to form a search query using either combinations of keywords or metacategories to search through a web page database containing text indices associated with one or more distinct web pages. After processing the user's request, therefore, the search engine returns a number of hits which correspond, generally, to URL pointers and text excerpts from the web pages that represent the closest match made by such search engine for the particular user query based on the search processing logic used by search engine. The structure and operation of such prior art search engines, including the mechanism by which they build the web page

database, and parse the search query, are well known in the art. To date, applicant is unaware of any such search engine that can easily and reliably search and retrieve information based on speech input from a user.

There are a number of reasons why the above environments (e-commerce, e-support, remote learning, INTERNET searching, etc.) do not utilize speech-based interfaces, despite the many benefits that would otherwise flow from such capability. First, there is obviously a requirement that the output of the speech recognizer be as accurate as possible. One of the more reliable approaches to speech recognition used at this time is based on the Hidden Markov Model (HMM)—a model used to mathematically describe any time series. A conventional usage of this technique is disclosed, for example, in U.S. Pat. No. 4,587,670 incorporated by reference herein. Because speech is considered to have an underlying sequence of one or more symbols, the HMM models corresponding to each symbol are trained on vectors from the speech waveforms. The Hidden Markov Model is a finite set of states, each of which is associated with a (generally multi-dimensional) probability distribution. Transitions among the states are governed by a set of probabilities called transition probabilities. In a particular state an outcome or observation can be generated, according to the associated probability distribution. This finite state machine changes state once every time unit, and each time t such that a state j is entered, a spectral parameter vector O_t is generated with probability density $B_j(O_t)$. It is only the outcome, not the state visible to an external observer and therefore states are "hidden" to the outside; hence the name Hidden Markov Model. The basic theory of HMMs was published in a series of classic papers by Baum and his colleagues in the late 1960's and early 1970's. HMMs were first used in speech applications by Baker at Carnegie Mellon, by Jelenik and colleagues at IBM in the late 1970's and by Steve Young and colleagues at Cambridge University, UK in the 1990's. Some typical papers and texts are as follows:

1. L. E. Baum, T. Petrie, "Statistical inference for probabilistic functions for finite state Markov chains", Ann. Math. Stat., 37: 1554–1563, 1966
2. L. E. Baum, "An inequality and associated maximation technique in statistical estimation for probabilistic functions of Markov processes", Inequalities 3: 1–8, 1972
3. J. H. Baker, "The dragon system—An Overview", IEEE Trans. on ASSP Proc., ASSP-23(1): 24–29, February 1975
4. F. Jeninek et al, "Continuous Speech Recognition: Statistical methods" in Handbook of Statistics, II, P. R. Kristnaiad, Ed. Amsterdam, The Netherlands, North-Holland, 1982
5. L. R. Bahl, F. Jeninek, R. L. Mercer, "A maximum likelihood approach to continuous speech recognition", IEEE Trans. Pattern Anal. Mach. Intell., PAMI-5: 179–190, 1983
6. J. D. Ferguson, "Hidden Markov Analysis: An Introduction", in Hidden Markov Models for Speech, Institute of Defense Analyses, Princeton, N.J. 1980.
7. H. R. Rabiner and B. H. Juang, "Fundamentals of Speech Recognition", Prentice Hall, 1993
8. H. R. Rabiner, "Digital Processing of Speech Signals", Prentice Hall, 1978

More recently research has progressed in extending HMM and combining HMMs with neural networks to speech recognition applications at various laboratories. The following is a representative paper:

9. Nelson Morgan, Hervé Bourlard, Steve Renals, Michael Cohen and Horacio Franco (1993), Hybrid Neural Network/Hidden Markov Model Systems for Continuous Speech Recognition. *Journal of Pattern Recognition and Artificial Intelligence*, Vol. 7, No. 4 pp. 899–916. Also in I. Guyon and P. Wang editors, *Advances in Pattern Recognition Systems using Neural Networks*, Vol. 7 of a Series in Machine Perception and Artificial Intelligence. World Scientific, February 1994.

All of the above are hereby incorporated by reference. While the HMM-based speech recognition yields very good results, contemporary variations of this technique cannot guarantee a word accuracy requirement of 100% exactly and consistently, as will be required for WWW applications for all possible all user and environment conditions. Thus, although speech recognition technology has been available for several years, and has improved significantly, the technical requirements have placed severe restrictions on the specifications for the speech recognition accuracy that is required for an application that combines speech recognition and natural language processing to work satisfactorily.

In contrast to word recognition, Natural language processing (NLP) is concerned with the parsing, understanding and indexing of transcribed utterances and larger linguistic units. Because spontaneous speech contains many surface phenomena such as disfluencies,—hesitations, repairs and restarts, discourse markers such as 'well' and other elements which cannot be handled by the typical speech recognizer, it is the problem and the source of the large gap that separates speech recognition and natural language processing technologies. Except for silence between utterances, another problem is the absence of any marked punctuation available for segmenting the speech input into meaningful units such as utterances. For optimal NLP performance, these types of phenomena should be annotated at its input. However, most continuous speech recognition systems produce only a raw sequence of words. Examples of conventional systems using NLP are shown in U.S. Pat. Nos. 4,991,094, 5,068,789, 5,146,405 and 5,680,628, all of which are incorporated by reference herein.

Second, most of the very reliable voice recognition systems are speaker-dependent, requiring that the interface be "trained" with the user's voice, which takes a lot of time, and is thus very undesirable from the perspective of a WWW environment, where a user may interact only a few times with a particular website. Furthermore, speaker-dependent systems usually require a large user dictionary (one for each unique user) which reduces the speed of recognition. This makes it much harder to implement a real-time dialog interface with satisfactory response capability (i.e., something that mirrors normal conversation—on the order of 3–5 seconds is probably ideal). At present, the typical shrink-wrapped speech recognition application software include offerings from IBM (VIAVOICE™) and Dragon Systems (DRAGON™). While most of these applications are adequate for dictation and other transcribing applications, they are woefully inadequate for applications such as NLQS where the word error rate must be close to 0%. In addition these offerings require long training times and are typically are non client-server configurations. Other types of trained systems are discussed in U.S. Pat. No. 5,231,670 assigned to Kurzweil, and which is also incorporated by reference herein.

Another significant problem faced in a distributed voice-based system is a lack of uniformity/control in the speech recognition process. In a typical stand-alone implementation of a speech recognition system, the entire SR engine runs on

a single client. A well-known system of this type is depicted in U.S. Pat. No. 4,991,217 incorporated by reference herein. These clients can take numerous forms (desktop PC, laptop PC, PDA, etc.) having varying speech signal processing and communications capability. Thus, from the server side perspective, it is not easy to assure uniform treatment of all users accessing a voice-enabled web page, since such users may have significantly disparate word recognition and error rate performances. While a prior art reference to Gould et al.—U.S. Pat. No. 5,915,236—discusses generally the notion of tailoring a recognition process to a set of available computational resources, it does not address or attempt to solve the issue of how to optimize resources in a distributed environment such as a client-server model. Again, to enable such voice-based technologies on a wide-spread scale it is far more preferable to have a system that harmonizes and accounts for discrepancies in individual systems so that even the thinnest client is supportable, and so that all users are able to interact in a satisfactory manner with the remote server running the e-commerce, e-support and/or remote learning application.

Two references that refer to a distributed approach for speech recognition include U.S. Pat. Nos. 5,956,683 and 5,960,399 incorporated by reference herein. In the first of these, U.S. Pat. No. 5,956,683—Distributed Voice Recognition System (assigned to Qualcomm) an implementation of a distributed voice recognition system between a telephony-based handset and a remote station is described. In this implementation, all of the word recognition operations seem to take place at the handset. This is done since the patent describes the benefits that result from locating of the system for acoustic feature extraction at the portable or cellular phone in order to limit degradation of the acoustic features due to quantization distortion resulting from the narrow bandwidth telephony channel. This reference therefore does not address the issue of how to ensure adequate performance for a very thin client platform. Moreover, it is difficult to determine, how, if at all, the system can perform real-time word recognition, and there is no meaningful description of how to integrate the system with a natural language processor.

The second of these references—U.S. Pat. No. 5,960,399—Client/Server Speech Processor/Recognizer (assigned to GTE) describes the implementation of a HMM-based distributed speech recognition system. This reference is not instructive in many respects, however, including how to optimize acoustic feature extraction for a variety of client platforms, such as by performing a partial word recognition process where appropriate. Most importantly, there is only a description of a primitive server-based recognizer that only recognizes the user's speech and simply returns certain keywords such as the user's name and travel destination to fill out a dedicated form on the user's machine. Also, the streaming of the acoustic parameters does not appear to be implemented in real-time as it can only take place after silence is detected. Finally, while the reference mentions the possible use of natural language processing (column **9**) there is no explanation of how such function might be implemented in a real-time fashion to provide an interactive feel for the user.

SUMMARY OF THE INVENTION

An object of the present invention, therefore, is to provide an improved system and method for overcoming the limitations of the prior art noted above;

A primary object of the present invention is to provide a word and phrase recognition system that is flexibly and optimally distributed across a client/platform computing architecture, so that improved accuracy, speed and uniformity can be achieved for a wide group of users;

A further object of the present invention is to provide a speech recognition system that efficiently integrates a distributed word recognition system with a natural language processing system, so that both individual words and entire speech utterances can be quickly and accurately recognized in any number of possible languages;

A related object of the present invention is to provide an efficient query response system so that an extremely accurate, real-time set of appropriate answers can be given in response to speech-based queries;

Yet another object of the present invention is to provide an interactive, real-time instructional/learning system that is distributed across a client/server architecture, and permits a real-time question/answer session with an interactive character;

A related object of the present invention is to implement such interactive character with an articulated response capability so that the user experiences a human-like interaction;

Still a further object of the present invention is to provide an INTERNET website with speech processing capability so that voice based data and commands can be used to interact with such site, thus enabling voice-based e-commerce and e-support services to be easily scaleable;

Another object is to implement a distributed speech recognition system that utilizes environmental variables as part of the recognition process to improve accuracy and speed;

A further object is to provide a scaleable query/response database system, to support any number of query topics and users as needed for a particular application and instantaneous demand;

Yet another object of the present invention is to provide a query recognition system that employs a two-step approach, including a relatively rapid first step to narrow down the list of potential responses to a smaller candidate set, and a second more computationally intensive second step to identify the best choice to be returned in response to the query from the candidate set;

A further object of the present invention is to provide a natural language processing system that facilitates query recognition by extracting lexical components of speech utterances, which components can be used for rapidly identifying a candidate set of potential responses appropriate for such speech utterances;

Another related object of the present invention is to provide a natural language processing system that facilitates query recognition by comparing lexical components of speech utterances with a candidate set of potential response to provide an extremely accurate best response to such query.

One general aspect of the present invention, therefore, relates to a natural language query system (NLQS) that offers a fully interactive method for answering user's questions over a distributed network such as the INTERNET or a local intranet. This interactive system when implemented over the worldwide web (WWW) services of the INTERNET functions so that a client or user can ask a question in a natural language such as English, French, German or Spanish and receive the appropriate answer at his or her personal computer also in his or her native natural language.

The system is distributed and consists of a set of integrated software modules at the client's machine and another

set of integrated software programs resident on a server or set of servers. The client-side software program is comprised of a speech recognition program, an agent and its control program, and a communication program. The server-side program is comprised of a communication program, a natural language engine (NLE), a database processor (DBProcess), an interface program for interfacing the DBProcess with the NLE, and a SQL database. In addition, the client's machine is equipped with a microphone and a speaker. Processing of the speech utterance is divided between the client and server side so as to optimize processing and transmission latencies, and so as to provide support for even very thin client platforms.

In the context of an interactive learning application, the system is specifically used to provide a single-best answer to a user's question. The question that is asked at the client's machine is articulated by the speaker and captured by a microphone that is built in as in the case of a notebook computer or is supplied as a standard peripheral attachment. Once the question is captured, the question is processed partially by NLQS client-side software resident in the client's machine. The output of this partial processing is a set of speech vectors that are transported to the server via the INTERNET to complete the recognition of the user's questions. This recognized speech is then converted to text at the server.

After the user's question is decoded by the speech recognition engine (SRE) located at the server, the question is converted to a structured query language (SQL) query. This query is then simultaneously presented to a software process within the server called DBProcess for preliminary processing and to a Natural Language Engine (NLE) module for extracting the noun phrases (NP) of the user's question. During the process of extracting the noun phrase within the NLE, the tokens of the users' question are tagged. The tagged tokens are then grouped so that the NP list can be determined. This information is stored and sent to the DBProcess process.

In the DBProcess, the SQL query is fully customized using the NP extracted from the user's question and other environment variables that are relevant to the application. For example, in a training application, the user's selection of course, chapter and or section would constitute the environment variables. The SQL query is constructed using the extended SQL Full-Text predicates —CONTAINS, FREETEXT, NEAR, AND. The SQL query is next sent to the Full-Text search engine within the SQL database, where a Full-Text search procedure is initiated. The result of this search procedure is recordset of answers. This recordset contains stored questions that are similar linguistically to the user's question. Each of these stored questions has a paired answer stored in a separate text file, whose path is stored in a table of the database.

The entire recordset of returned stored answers is then returned to the NLE engine in the form of an array. Each stored question of the array is then linguistically processed sequentially one by one. This linguistic processing constitutes the second step of a 2-step algorithm to determine the single best answer to the user's question. This second step proceeds as follows: for each stored question that is returned in the recordset, a NP of the stored question is compared with the NP of the user's question. After all stored questions of the array are compared with the user's question, the stored question that yields the maximum match with the user's question is selected as the best possible stored question that matches the user's question. The metric that is used to determine the best possible stored question is the number of noun phrases.

The stored answer that is paired to the best-stored question is selected as the one that answers the user's question. The ID tag of the question is then passed to the DBProcess. This DBProcess returns the answer which is stored in a file.

A communication link is again established to send the answer back to the client in compressed form. The answer once received by the client is decompressed and articulated to the user by the text-to-speech engine. Thus, the invention can be used in any number of different applications involving interactive learning systems, INTERNET related commerce sites, INTERNET search engines, etc.

Computer-assisted instruction environments often require the assistance of mentors or live teachers to answer questions from students. This assistance often takes the form of organizing a separate pre-arranged forum or meeting time that is set aside for chat sessions or live call-in sessions so that at a scheduled time answers to questions may be provided. Because of the time immediacy and the on-demand or asynchronous nature of on-line training where a student may log on and take instruction at any time and at any location, it is important that answers to questions be provided in a timely and cost-effective manner so that the user or student can derive the maximum benefit from the material presented.

This invention addresses the above issues. It provides the user or student with answers to questions that are normally channeled to a live teacher or mentor. This invention provides a single-best answer to questions asked by the student. The student asks the question in his or her own voice in the language of choice. The speech is recognized and the answer to the question is found using a number of technologies including distributed speech recognition, full-text search database processing, natural language processing and text-to-speech technologies. The answer is presented to the user, as in the case of a live teacher, in an articulated manner by an agent that mimics the mentor or teacher, and in the language of choice—English, French, German, Japanese or other natural spoken language. The user can choose the agent's gender as well as several speech parameters such as pitch, volume and speed of the character's voice.

Other applications that benefit from NLQS are e-commerce applications. In this application, the user's query for a price of a book, compact disk or for the availability of any item that is to be purchased can be retrieved without the need to pick through various lists on successive web pages. Instead, the answer is provided directly to the user without any additional user input.

Similarly, it is envisioned that this system can be used to provide answers to frequently-asked questions (FAQs), and as a diagnostic service tool for e-support. These questions are typical of a give web site and are provided to help the user find information related to a payment procedure or the specifications of, or problems experienced with a product/service. In all of these applications, the NLQS architecture can be applied.

A number of inventive methods associated with these architectures are also beneficially used in a variety of INTERNET related applications.

Although the inventions are described below in a set of preferred embodiments, it will be apparent to those skilled in the art the present inventions could be beneficially used in many environments where it is necessary to implement fast, accurate speech recognition, and/or to provide a human-like dialog capability to an intelligent system.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram of a preferred embodiment of a natural language query system (NLQS) of the present invention, which is distributed across a client/server computing architecture, and can be used as an interactive learning system, an e-commerce system, an e-support system, and the like;

FIG. **2** is a block diagram of a preferred embodiment of a client side system, including speech capturing modules, partial speech processing modules, encoding modules, transmission modules, agent control modules, and answer/voice feedback modules that can be used in the aforementioned NLQS;

FIG. **2**—**2** is a block diagram of a preferred embodiment of a set of initialization routines and procedures used for the client side system of FIG. **2**;

FIG. **3** is a block diagram of a preferred embodiment of a set of routines and procedures used for handling an iterated set of speech utterances on the client side system of FIG. **2**, transmitting speech data for such utterances to a remote server, and receiving appropriate responses back from such server;

FIG. **4** is a block diagram of a preferred embodiment of a set of initialization routines and procedures used for un-initializing the client side system of FIG. **2**;

FIG. **4**A is a block diagram of a preferred embodiment of a set of routines and procedures used for implementing a distributed component of a speech recognition module for the server side system of FIG. **5**;

FIG. **4**B is a block diagram of a preferred set of routines and procedures used for implementing an SQL query builder for the server side system of FIG. **5**;

FIG. **4**C is a block diagram of a preferred embodiment of a set of routines and procedures used for implementing a database control process module for the server side system of FIG. **5**;

FIG. **4**D is a block diagram of a preferred embodiment of a set of routines and procedures used for implementing a natural language engine that provides query formulation support, a query response module, and an interface to the database control process module for the server side system of FIG. **5**;

FIG. **5** is a block diagram of a preferred embodiment of a server side system, including a speech recognition module to complete processing of the speech utterances, environmental and grammar control modules, query formulation modules, a natural language engine, a database control module, and a query response module that can be used in the aforementioned NLQS;

FIG. **6** illustrates the organization of a full-text database used as part of server side system shown in FIG. **5**;

FIG. **7**A illustrates the organization of a full-text database course table used as part of server side system shown in FIG. **5** for an interactive learning embodiment of the present invention;

FIG. **7**B illustrates the organization of a full-text database chapter table used as part of server side system shown in FIG. **5** for an interactive learning embodiment of the present invention;

FIG. **7**C describes the fields used in a chapter table used as part of server side system shown in FIG. **5** for an interactive learning embodiment of the present invention;

FIG. **7**D describes the fields used in a section table used as part of server side system shown in FIG. **5** for an interactive learning embodiment of the present invention;

FIG. **8** is a flow diagram of a first set of operations performed by a preferred embodiment of a natural language engine on a speech utterance including Tokenization, Tagging and Grouping;

FIG. **9** is a flow diagram of the operations performed by a preferred embodiment of a natural language engine on a speech utterance including stemming and Lexical Analysis

FIG. **10** is a block diagram of a preferred embodiment of a SQL database search and support system for the present invention;

FIGS. **11**A–**11**C are flow diagrams illustrating steps performed in a preferred two step process implemented for query recognition by the NLQS of FIG. **2**;

FIG. **12** is an illustration of another embodiment of the present invention implemented as part of a Web-based speech based learning/training System;

FIGS. **13**–**17** are illustrations of another embodiment of the present invention implemented as part of a Web-based e-commerce system;

FIG. **18** is an illustration of another embodiment of the present invention implemented as part of a voice-based Help Page for an E-Commerce Web Site.

DETAILED DESCRIPTION OF THE INVENTION

Overview

As alluded to above, the present inventions allow a user to ask a question in a natural language such as English, French, German, Spanish or Japanese at a client computing system (which can be as simple as a personal digital assistant or cell-phone, or as sophisticated as a high end desktop PC) and receive an appropriate answer from a remote server also in his or her native natural language. As such, the embodiment of the invention shown in FIG. **1** is beneficially used in what can be generally described as a Natural Language Query System (NLQS) **100**, which is configured to interact on a real-time basis to give a human-like dialog capability/experience for e-commerce, e-support, and e-learning applications.

The processing for NLQS **100** is generally distributed across a client side system **150**, a data link **160**, and a server-side system **180**. These components are well known in the art, and in a preferred embodiment include a personal computer system **150**, an INTERNET connection **160**A, **160**B, and a larger scale computing system **180**. It will be understood by those skilled in the art that these are merely exemplary components, and that the present invention is by no means limited to any particular implementation or combination of such systems. For example, client-side system **150** could also be implemented as a computer peripheral, a PDA, as part of a cell-phone, as part of an INTERNET-adapted appliance, an INTERNET linked kiosk, etc. Similarly, while an INTERNET connection is depicted for data link **160**A, it is apparent that any channel that is suitable for carrying data between client system **150** and server system **180** will suffice, including a wireless link, an RF link, an IR link, a LAN, and the like. Finally, it will be further appreciated that server system **180** may be a single, large-scale system, or a collection of smaller systems interlinked to support a number of potential network users.

Initially speech input is provided in the form of a question or query articulated by the speaker at the client's machine or personal accessory as a speech utterance. This speech utterance is captured and partially processed by NLQS client-side software **155** resident in the client's machine. To

facilitate and enhance the human-like aspects of the interaction, the question is presented in the presence of an animated character **157** visible to the user who assists the user as a personal information retriever/agent. The agent can also interact with the user using both visible text output on a monitor/display (not shown) and/or in audible form using a text to speech engine **159**. The output of the partial processing done by SRE **155** is a set of speech vectors that are transmitted over communication channel **160** that links the user's machine or personal accessory to a server or servers via the INTERNET or a wireless gateway that is linked to the INTERNET as explained above. At server **180**, the partially processed speech signal data is handled by a server-side SRE **182**, which then outputs recognized speech text corresponding to the user's question. Based on this user question related text, a text-to-query converter **184** formulates a suitable query that is used as input to a database processor **186**. Based on the query, database processor **186** then locates and retrieves an appropriate answer using a customized SQL query from database **188**. A Natural Language Engine **190** facilitates structuring the query to database **188**. After a matching answer to the user's question is found, the former is transmitted in text form across data link **160**B, where it is converted into speech by text to speech engine **159**, and thus expressed as oral feedback by animated character agent **157**.

Because the speech processing is broken up in this fashion, it is possible to achieve real-time, interactive, human-like dialog consisting of a large, controllable set of questions/answers. The assistance of the animated agent **157** further enhances the experience, making it more natural and comfortable for even novice users. To make the speech recognition process more reliable, context-specific grammars and dictionaries are used, as well as natural language processing routines at NLE **190**, to analyze user questions lexically. While context-specific processing of speech data is known in the art (see e.g., U.S. Pat. Nos. 5,960,394, 5,867,817, 5,758,322 and 5,384,892 incorporated by reference herein) the present inventors are unaware of any such implementation as embodied in the present inventions. The text of the user's question is compared against text of other questions to identify the question posed by the user by DB processor/engine (DBE) **186**. By optimizing the interaction and relationship of the SR engines **155** and **182**, the NLP routines **190**, and the dictionaries and grammars, an extremely fast and accurate match can be made, so that a unique and responsive answer can be provided to the user.

On the server side **180**, interleaved processing further accelerates the speech recognition process. In simplified terms, the query is presented simultaneously both to NLE **190** after the query is formulated, as well as to DBE **186**. NLE **190** and SRE **182** perform complementary functions in the overall recognition process. In general, SRE **182** is primarily responsible for determining the identity of the words articulated by the user, while NLE **190** is responsible for the linguistic morphological analysis of both the user's query and the search results returned after the database query.

After the user's query is analyzed by NLE **190** some parameters are extracted and sent to the DBProcess. Additional statistics are stored in an array for the 2^{nd} step of processing. During the 2^{nd} step of 2-step algorithm, the recordset of preliminary search results are sent to the NLE **160** for processing. At the end of this 2^{nd} step, the single question that matches the user's query is sent to the DBProcess where further processing yields the paired answer that is paired with the single best stored question.

Thus, the present invention uses a form of natural language processing (NLP) to achieve optimal performance in a speech based web application system. While NLP is known in the art, prior efforts in Natural Language Processing (NLP) work nonetheless have not been well integrated with Speech Recognition (SR) technologies to achieve reasonable results in a web-based application environment. In speech recognition, the result is typically a lattice of possible recognized words each with some probability of fit with the speech recognizer. As described before, the input to a typical NLP system is typically a large linguistic unit. The NLP system is then charged with the parsing, understanding and indexing of this large linguistic unit or set of transcribed utterances. The result of this NLP process is to understand lexically or morphologically the entire linguistic unit as opposed to word recognition. Put another way, the linguistic unit or sentence of connected words output by the SRE has to be understood lexically, as opposed to just being "recognized".

As indicated earlier, although speech recognition technology has been available for several years, the technical requirements for the NLQS invention have placed severe restrictions on the specifications for the speech recognition accuracy that is required for an application that combines speech recognition and natural language processing to work satisfactorily. In realizing that even with the best of conditions, it might be not be possible to achieve the perfect 100% speech recognition accuracy that is required, the present invention employs an algorithm that balances the potential risk of the speech recognition process with the requirements of the natural language processing so that even in cases where perfect speech recognition accuracy is not achieved for each word in the query, the entire query itself is nonetheless recognized with sufficient accuracy.

This recognition accuracy is achieved even while meeting very stringent user constraints, such as short latency periods of 3 to 5 seconds (ideally—ignoring transmission latencies which can vary) for responding to a speech-based query, and for a potential set of 100–250 query questions. This quick response time gives the overall appearance and experience of a real-time discourse that is more natural and pleasant from the user's perspective. Of course, non-real time applications, such as translation services for example, can also benefit from the present teachings as well, since a centralized set of HMMs, grammars, dictionaries, etc., are maintained.

General Aspects of Speech Recognition Used in the Present Inventions

General background information on speech recognition can be found in the prior art references discussed above and incorporated by reference herein. Nonetheless, a discussion of some particular exemplary forms of speech recognition structures and techniques that are well-suited for NLQS **100** is provided next to better illustrate some of the characteristics, qualities and features of the present inventions.

Speech recognition technology is typically of two types—speaker independent and speaker dependent. In speaker-dependent speech recognition technology, each user has a voice file in which a sample of each potentially recognized word is stored. Speaker-dependent speech recognition systems typically have large vocabularies and dictionaries making them suitable for applications as dictation and text transcribing. It follows also that the memory and processor resource requirements for the speaker-dependent can be and are typically large and intensive.

Conversely speaker-independent speech recognition technology allows a large group of users to use a single vocabulary file. It follows then that the degree of accuracy that can be achieved is a function of the size and complexity of the grammars and dictionaries that can be supported for a given language. Given the context of applications for which NLQS, the use of small grammars and dictionaries allow speaker independent speech recognition technology to be implemented in NLQS.

The key issues or requirements for either type—speaker-independent or speaker-dependent, are accuracy and speed. As the size of the user dictionaries increase, the speech recognition accuracy metric—word error rate (WER) and the speed of recognition decreases. This is so because the search time increases and the pronunciation match becomes more complex as the size of the dictionary increases.

The basis of the NLQS speech recognition system is a series of Hidden Markov Models (HMM), which, as alluded to earlier, are mathematical models used to characterize any time varying signal. Because parts of speech are considered to be based on an underlying sequence of one or more symbols, the HMM models corresponding to each symbol are trained on vectors from the speech waveforms. The Hidden Markov Model is a finite set of states, each of which is associated with a (generally multi-dimensional) probability distribution. Transitions among the states are governed by a set of probabilities called transition probabilities. In a particular state an outcome or observation can be generated, according to an associated probability distribution. This finite state machine changes state once every time unit, and each time t such that a state j is entered, a spectral parameter vector O_t is generated with probability density $B_j(O_t)$. It is only the outcome, not the state which is visible to an external observer and therefore states are "hidden" to the outside; hence the name Hidden Markov Model.

In isolated speech recognition, it is assumed that the sequence of observed speech vectors corresponding to each word can each be described by a Markov model as follows:

$$O=o_1, o_2, \ldots o_T \quad (1\text{-}1)$$

where o_t is a speech vector observed at time t. The isolated word recognition then is to compute:

$$\arg\max\{P(w_i|O)\} \quad (1\text{-}2)$$

By using Bayes' Rule,

$$\{P(w_i|O)\}=[P(O|w_i)P(w_i)]/P(O) \quad (1\text{-}3)$$

In the general case, the Markov model when applied to speech also assumes a finite state machine which changes state once every time unit and each time that a state j is entered, a speech vector o_t is generated from the probability density $b_j(o_t)$. Furthermore, the transition from state i to state j is also probabilistic and is governed by the discrete probability a_{ij}.

For a state sequence X, the joint probability that O is generated by the model M moving through a state sequence X is the product of the transition probabilities and the output probabilities. Only the observation sequence is known—the state sequence is hidden as mentioned before.

Given that X is unknown, the required likelihood is computed by summing over all possible state sequences X=x(1), x(2), x(3), . . . x(T), that is

$$P(O|M)=\Sigma\{a_{x(0)\,x(1)}\Pi b(x)(o_t)a_{x(t)\,x(t+1)}\}$$

Given a set of models M_i, corresponding to words w_i equation 1-2 is solved by using 1-3 and also by assuming that:

$$P(O|w_i)=P(O|M_i)$$

All of this assumes that the parameters $\{a_{ij}\}$ and $\{b_j(o_t)\}$ are known for each model M_i. This can be done, as explained earlier, by using a set of training examples corresponding to a particular model. Thereafter, the parameters of that model can be determined automatically by a robust and efficient re-estimation procedure. So if a sufficient number of representative examples of each word are collected, then a HMM can be constructed which simply models all of the many sources of variability inherent in real speech. This training is well-known in the art, so it is not described at length herein, except to note that the distributed architecture of the present invention enhances the quality of HMMs, since they are derived and constituted at the server side, rather than the client side. In this way, appropriate samples from users of different geographical areas can be easily compiled and analyzed to optimize the possible variations expected to be seen across a particular language to be recognized. Uniformity of the speech recognition process is also well-maintained, and error diagnostics are simplified, since each prospective user is using the same set of HMMs during the recognition process.

To determine the parameters of a HMM from a set of training samples, the first step typically is to make a rough guess as to what they might be. Then a refinement is done using the Baum-Welch estimation formulae. By these formulae, the maximum likelihood estimates of μ_j (where μ_j is mean vector and Σ_j is covariance matrix) is:

$$\mu_j=\Sigma_{t=1}^{T} L_j(t)o_t/[\Sigma_{t=1}^{T} L_j(t)o_t]$$

A forward-backward algorithm is next used to calculate the probability of state occupation $L_j(t)$. If the forward probability $\alpha_j(t)$ for some model M with N states is defined as:

$$\alpha_j(t)=P(o_1, \ldots, o_t, x(t)=j|M)$$

This probability can be calculated using the recursion:

$$\alpha_j(t)=[\Sigma^{N-1}_{i=2}\alpha(t-1)a_{ij}]b_j(o_t)$$

Similarly the backward probability can be computed using the recursion:

$$\beta_j(t)=\Sigma^{N-1}_{j=2}a_{ij}b_j(o_t+1)(t+1)$$

Realizing that the forward probability is a joint probability and the backward probability is a conditional probability, the probability of state occupation is the product of the two probabilities:

$$\alpha j(t)\beta_j(t)=P(O, x(t)=j|M)$$

Hence the probability of being in state j at a time t is:

$$L_j(t)=1/P[\alpha_j(t)\beta_j(t)]$$

where P=P(O|M)

To generalize the above for continuous speech recognition, we assume the maximum likelihood state sequence where the summation is replaced by a maximum operation. Thus for a given model M, let ϕj (t) represent the maximum likelihood of observing speech vectors o_1 to o_t and being used in state j at time t:

$$\phi_j(t)=\max\{\phi j(t)(t-1)\alpha_{ij}\}\beta_j(o_t)$$

Expressing this logarithmically to avoid underflow, this likelihood becomes:

$$\psi_j(t)=\max\{\psi_t(t-1)+\log(\alpha_{ij})\}+\log(b_j(o_t)$$

This is also known as the Viterbi algorithm. It can be visualized as finding the best path through a matrix where the vertical dimension represents the states of the HMM and

horizontal dimension represents frames of speech i.e. time. To complete the extension to connected speech recognition, it is further assumed that each HMM representing the underlying sequence is connected. Thus the training data for continuous speech recognition should consist of connected utterances; however, the boundaries between words do not have to be known.

To improve computational speed/efficiency, the Viterbi algorithm is sometimes extended to achieve convergence by using what is known as a Token Passing Model. The token passing model represents a partial match between the observation sequence o_1 to o_t and a particular model, subject to the constraint that the model is in state j at time t. This token passing model can be extended easily to connected speech environments as well if we allow the sequence of HMMs to be defined as a finite state network. A composite network that includes both phoneme-based HMMs and complete words can be constructed so that a single-best word can be recognized to form connected speech using word N-best extraction from the lattice of possibilities. This composite form of HMM-based connected speech recognizer is the basis of the NLQS speech recognizer module. Nonetheless, the present invention is not limited as such to such specific forms of speech recognizers, and can employ other techniques for speech recognition if they are otherwise compatible with the present architecture and meet necessary performance criteria for accuracy and speed to provide a real-time dialog experience for users.

The representation of speech for the present invention's HMM-based speech recognition system assumes that speech is essentially either a quasi-periodic pulse train (for voiced speech sounds) or a random noise source (for unvoiced sounds). It may be modeled as two sources—one a impulse train generator with pitch period P and a random noise generator which is controlled by a voice/unvoiced switch. The output of the switch is then fed into a gain function estimated from the speech signal and scaled to feed a digital filter H(z) controlled by the vocal tract parameter characteristics of the speech being produced. All of the parameters for this model—the voiced/unvoiced switching, the pitch period for voiced sounds, the gain parameter for the speech signal and the coefficient of the digital filter, vary slowly with time. In extracting the acoustic parameters from the user's speech input so that it can evaluated in light of a set of HMMs, cepstral analysis is typically used to separate the vocal tract information from the excitation information. The cepstrum of a signal is computed by taking the Fourier (or similar) transform of the log spectrum. The principal advantage of extracting cepstral coefficients is that they are de-correlated and the diagonal covariances can be used with HMMs. Since the human ear resolves frequencies non-linearly across the audio spectrum, it has been shown that a front-end that operates in a similar non-linear way improves speech recognition performance.

Accordingly, instead of a typical linear prediction-based analysis, the front-end of the NLQS speech recognition engine implements a simple, fast Fourier transform based filter bank designed to give approximately equal resolution on the Mel-scale. To implement this filter bank, a window of speech data (for a particular time frame) is transformed using a software based Fourier transform and the magnitude taken. Each FFT magnitude is then multiplied by the corresponding filter gain and the results accumulated. The cepstral coefficients that are derived from this filter-bank analysis at the front end are calculated during a first partial processing phase of the speech signal by using a Discrete Cosine Transform of the log filter bank amplitudes. These cepstral coefficients are called Mel-Frequency Cepstral Coefficients (MFCC) and they represent some of the speech parameters transferred from the client side to characterize the acoustic features of the user's speech signal. These parameters are chosen for a number of reasons, including the fact that they can be quickly and consistently derived even across systems of disparate capabilities (i.e., for everything from a low power PDA to a high powered desktop system), they give good discrimination, they lend themselves to a number of useful recognition related manipulations, and they are relatively small and compact in size so that they can be transported rapidly across even a relatively narrow band link. Thus, these parameters represent the least amount of information that can be used by a subsequent server side system to adequately and quickly complete the recognition process.

To augment the speech parameters an energy term in the form of the logarithm of the signal energy is added. Accordingly, RMS energy is added to the 12 MFCC's to make 13 coefficients. These coefficients together make up the partially processed speech data transmitted in compressed form from the user's client system to the remote server side.

The performance of the present speech recognition system is enhanced significantly by computing and adding time derivatives to the basic static MFCC parameters at the server side. These two other sets of coefficients—the delta and acceleration coefficients representing change in each of the 13 values from frame to frame (actually measured across several frames), are computed during a second partial speech signal processing phase to complete the initial processing of the speech signal, and are added to the original set of coefficients after the latter are received. These MFCCs together with the delta and acceleration coefficients constitute the observation vector O_t mentioned above that is used for determining the appropriate HMM for the speech data.

The delta and acceleration coefficients are computed using the following regression formula:

$$d_t=\Sigma^{\theta}_{\theta=1}[c_{t+\theta}-c_{t-\theta}]/2\Sigma^{\theta}_{\theta=1}\theta^2$$

where d_t is a delta coefficient at time t computed in terms of the corresponding static coefficients:

$$d_t=[c_{t+\theta}-c_{t-\theta}]/2\theta$$

In a typical stand-alone implementation of a speech recognition system, the entire SR engine runs on a single client. In other words, both the first and second partial processing phases above are executed by the same DSP (or microprocessor) running a ROM or software code routine at the client's computing machine.

In contrast, because of several considerations, specifically—cost, technical performance, and client hardware uniformity, the present NLQS system uses a partitioned or distributed approach. While some processing occurs on the client side, the main speech recognition engine runs on a centrally located server or number of servers. More specifically, as noted earlier, capture of the speech signals, MFCC vector extraction and compression are implemented on the client's machine during a first partial processing phase. The routine is thus streamlined and simple enough to be implemented within a browser program (as a plug in module, or a downloadable applet for example) for maximum ease of use and utility. Accordingly, even very "thin" client platforms can be supported, which enables the use of the present system across a greater number of potential sites. The primary MFCCs are then transmitted to the server over the channel, which, for example, can include a dial-up INTERNET connection, a LAN connection, a wireless connection

and the like. After decompression, the delta and acceleration coefficients are computed at the server to complete the initial speech processing phase, and the resulting observation vectors O_t are also determined.

General Aspects of Speech Recognition Engine

The speech recognition engine is also located on the server, and is based on a HTK-based recognition network compiled from a word-level network, a dictionary and a set of HMMs. The recognition network consists of a set of nodes connected by arcs. Each node is either a HMM model instance or a word end. Each model node is itself a network consisting of states connected by arcs. Thus when fully compiled, a speech recognition network consists of HMM states connected by transitions. For an unknown input utterance with T frames, every path from the start node to the exit node of the network passes through T HMM states. Each of these paths has log probability which is computed by summing the log probability of each individual transition in the path and the log probability of each emitting state generating the corresponding observation. The function of the Viterbi decoder is find those paths through the network which have the highest log probability. This is found using the Token Passing algorithm. In a network that has many nodes, the computation time is reduced by only allowing propagation of those tokens which will have some chance of becoming winners. This process is called pruning.

Natural Language Processor

In a typical natural language interface to a database, the user enters a question in his/her natural language, for example, English. The system parses it and translates it to a query language expression. The system then uses the query language expression to process the query and if the search is successful, a recordset representing the results is displayed in English either formatted as raw text or in a graphical form. For a natural language interface to work well involves a number of technical requirements.

For example, it needs to be robust—in the sentence 'What's the departments turnover' it needs to decide that the word whats=what's=what is. And it also has to determine that departments=department's. In addition to being robust, the natural language interface has to distinguish between the several possible forms of ambiguity that may exist in the natural language—lexical, structural, reference and ellipsis ambiguity. All of these requirements, in addition to the general ability to perform basic linguistic morphological operations of tokenization, tagging and grouping, are implemented within the present invention.

Tokenization is implemented by a text analyzer which treats the text as a series of tokens or useful meaningful units that are larger than individual characters, but smaller than phrases and sentences. These include words, separable parts of words, and punctuation. Each token is associated with an offset and a length. The first phase of tokenization is the process of segmentation which extracts the individual tokens from the input text and keeps track of the offset where each token originated in the input text. The tokenizer output lists the offset and category for each token. In the next phase of the text analysis, the tagger uses a built-in morphological analyzer to look up each word/token in a phrase or sentence and internally lists all parts of speech. The output is the input string with each token tagged with a parts of speech notation. Finally the grouper which functions as a phrase extractor or phrase analyzer, determines which groups of words form phrases. These three operations which are the foundations for any modern linguistic processing schemes, are fully implemented in optimized algorithms for determining the single-best possible answer to the user's question.

SQL Database and Full-Text Query

Another key component of present system is a SQL-database. This database is used to store text, specifically the answer-question pairs are stored in full-text tables of the database. Additionally, the full-text search capability of the database allows full-text searches to be carried out.

While a large portion of all digitally stored information is in the form of unstructured data, primarily text, it is now possible to store this textual data in traditional database systems in character-based columns such as varchar and text. In order to effectively retrieve textual data from the database, techniques have to be implemented to issue queries against textual data and to retrieve the answers in a meaningful way where it provides the answers as in the case of the NLQS system.

There are two major types of textual searches: Property—This search technology first applies filters to documents in order to extract properties such as author, subject, type, word count, printed page count, and time last written, and then issues searches against those properties; Full-text—this search technology first creates indexes of all non-noise words in the documents, and then uses these indexes to support linguistic searches and proximity searches.

Two additional technologies are also implemented in this particular RDBMs: SQL Server also have been integrated: A Search service—a full-text indexing and search service that is called both index engine and search, and a parser that accepts full-text SQL extensions and maps them into a form that can be processed by the search engine.

The four major aspects involved in implementing full-text retrieval of plain-text data from a full-text-capable database are: Managing the definition of the tables and columns that are registered for full-text searches; Indexing the data in registered columns—the indexing process scans the character streams, determines the word boundaries (this is called word breaking), removes all noise words (this also is called stop words), and then populates a full-text index with the remaining words; Issuing queries against registered columns for populated full-text indexes; Ensuring that subsequent changes to the data in registered columns gets propagated to the index engine to keep the full-text indexes synchronized.

The underlying design principle for the indexing, querying, and synchronizing processes is the presence of a full-text unique key column (or single-column primary key) on all tables registered for full-text searches. The full-text index contains an entry for the non-noise words in each row together with the value of the key column for each row.

When processing a full-text search, the search engine returns to the database the key values of the rows that match the search criteria.

The full-text administration process starts by designating a table and its columns of interest for full-text search. Customized NLQS stored procedures are used first to register tables and columns as eligible for full-text search. After that, a separate request by means of a stored procedure is issued to populate the full-text indexes. The result is that the underlying index engine gets invoked and asynchronous index population begins. Full-text indexing tracks which significant words are used and where they are located. For example, a full-text index might indicate that the word "NLQS" is found at word number **423** and word number **982** in the Abstract column of the DevTools table for the row associated with a ProductID of 6. This index structure supports an efficient search for all items containing indexed

words as well as advanced search operations, such as phrase searches and proximity searches. (An example of a phrase search is looking for "white elephant," where "white" is followed by "elephant". An example of a proximity search is looking for "big" and "house" where "big" occurs near "house".) To prevent the full-text index from becoming bloated, noise words such as "a," "and," and "the" are ignored.

Extensions to the Transact-SQL language are used to construct full-text queries. The two key predicates that are used in the NLQS are CONTAINS and FREETEXT.

The CONTAINS predicate is used to determine whether or not values in full-text registered columns contain certain words and phrases. Specifically, this predicate is used to search for:

A word or phrase.

The prefix of a word or phrase.

A word or phrase that is near another.

A word that is an inflectional form of another (for example, "drive" is the inflectional stem of "drives," "drove," "driving," and "driven").

A set of words or phrases, each of which is assigned a different weighting.

The relational engine within SQL Server recognizes the CONTAINS and FREETEXT predicates and performs some minimal syntax and semantic checking, such as ensuring that the column referenced in the predicate has been registered for full-text searches. During query execution, a full-text predicate and other relevant information are passed to the full-text search component. After further syntax and semantic validation, the search engine is invoked and returns the set of unique key values identifying those rows in the table that satisfy the full-text search condition. In addition to the FREETEXT and CONTAINS, other predicates such as AND, LIKE, NEAR are combined to create the customized NLQS SQL construct.

Full-Text Query Architecture of the SQL Database

The full-text query architecture is comprised of the following several components—Full-Text Query component, the SQL Server Relational Engine, the Full-Text provider and the Search Engine.

The Full-Text Query component of the SQL database accept a full-text predicate or rowset-valued function from the SQL Server; transform parts of the predicate into an internal format, and sends it to Search Service, which returns the matches in a rowset. The rowset is then sent back to SQL Server. SQL Server uses this information to create the resultset that is then returned to the submitter of the query.

The SQL Server Relational Engine accepts the CONTAINS and FREETEXT predicates as well as the CONTAINSTABLE( ) and FREETEXTTABLE( ) rowset-valued functions. During parse time, this code checks for conditions such as attempting to query a column that has not been registered for full-text search. If valid, then at run time, the ft_search_condition and context information is sent to the full-text provider. Eventually, the full-text provider returns a rowset to SQL Server, which is used in any joins (specified or implied) in the original query. The Full-Text Provider parses and validates the ft_search_condition, constructs the appropriate internal representation of the full-text search condition, and then passes it to the search engine. The result is returned to the relational engine by means of a rowset of rows that satisfy ft_search_condition.

Client Side System **150**

The architecture of client-side system **150** of Natural Language Query System **100** is illustrated in greater detail in FIG. **2**. Referring to FIG. **2**, the three main processes effectuated by Client System **150** are illustrated as follows: Initialization process **200**A consisting of SRE **201**, Communication **202** and Microsoft (MS) Agent **203** routines; an iterative process **200**B consisting of two sub-routines: a) Receive User Speech **208**—made up of SRE **204** and Communication **205**; and b) Receive Answer from Server **207**—made up of MS Speak Agent **206**, Communication **209**, Voice data file **210** and Text to Speech Engine **211**. Finally, un-initialization process **200**C is made up of three sub-routines: SRE **212**, Communication **213**, and MS Agent **214**. Each of the above three processes are described in detail in the following paragraphs. It will be appreciated by those skilled in the art that the particular implementation for such processes and routines will vary from client platform to platform, so that in some environments such processes may be embodied in hard-coded routines executed by a dedicated DSP, while in others they may be embodied as software routines executed by a shared host processor, and in still others a combination of the two may be used.

Initialization at Client System **150**

The initialization of the Client System **150** is illustrated in FIG. **2**—**2** and is comprised generally of 3 separate initializing processes: client-side Speech Recognition Engine **220**A, MS Agent **220**B and Communication processes **220**C.

Initialization of Speech Recognition Engine **220**A

Speech Recognition Engine **155** is initialized and configured using the routines shown in **220**A. First, an SRE COM Library is initialized. Next, memory **220** is allocated to hold Source and Coder objects, are created by a routine **221**. Loading of configuration file **221**A from configuration data file **221**B also takes place at the same time that the SRE Library is initialized. In configuration file **221**B, the type of the input of Coder and the type of the output of the Coder are declared. The structure, operation, etc. of such routines are well-known in the art, and they can be implemented using a number of fairly straightforward approaches. Accordingly, they are not discussed in detail herein. Next, Speech and Silence components of an utterance are calibrated using a routine **222**, in a procedure that is also well-known in the art. To calibrate the speech and silence components, the user preferably articulates a sentence that is displayed in a text box on the screen. The SRE library then estimates the noise and other parameters required to find e silence and speech elements of future user utterances.

Initialization of MS Agent **220**B

The software code used to initialize and set up a MS Agent **220**B is also illustrated in FIG. **2**—**2**. The MS Agent **220**B routine is responsible for coordinating and handling the actions of the animated agent **157** (FIG. **1**). This initialization thus consists of the following steps:

1. Initialize COM library **223**. This part of the code initializes the COM library, which is required to use ActiveX Controls, which controls are well-known in the art.
2. Create instance of Agent Server **224**—this part of the code creates an instance of Agent ActiveX control.
3. Loading of MS Agent **225**—this part of the code loads MS Agent character from a specified file **225**A containing general parameter data for the Agent Character, such as the overall appearance, shape, size, etc.
4. Get Character Interface **226**—this part of the code gets an appropriate interface for the specified character; for

example, characters may have different control/interaction capabilities that can be presented to the user.

5. Add Commands to Agent Character Option **227**—this part of the code adds commands to an Agent Properties sheet, which sheet can be accessed by clicking on the icon that appears in the system tray, when the Agent character is loaded e.g., that the character can Speak, how he/she moves, TTS Properties, etc.
6. Show the Agent Character **228**—this part of the code displays the Agent character on the screen so it can be seen by the user;
7. AgentNotifySink—to handle events. This part of the code creates AgentNotifySink object **229**, registers it at **230** and then gets the Agent Properties interface **231**. The property sheet for the Agent character is assigned using routine **232**.
8. Do Character Animations **233**—This part of the code plays specified character animations to welcome the user to NLQS **100**.

The above then constitutes the entire sequence required to initialize the MS Agent. As with the SRE routines, the MS Agent routines can be implemented in any suitable and conventional fashion by those skilled in the art based on the present teachings. The particular structure, operation, etc. of such routines is not critical, and thus they are not discussed in detail herein.

In a preferred embodiment, the MS Agent is configured to have an appearance and capabilities that are appropriate for the particular application. For instance, in a remote learning application, the agent has the visual form and mannerisms/attitude/gestures of a college professor. Other visual props (blackboard, textbook, etc.) may be used by the agent and presented to the user to bring to mind the experience of being in an actual educational environment. The characteristics of the agent may be configured at the client side **150**, and/or as part of code executed by a browser program (not shown) in response to configuration data and commands from a particular web page. For example, a particular website offering medical services may prefer to use a visual image of a doctor. These and many other variations will be apparent to those skilled in the art for enhancing the human-like, real-time dialog experience for users.

Initialization of Communication Link **160**A

The initialization of Communication Link **160**A is shown with reference to process **220**C FIG. **2**—**2**. Referring to FIG. **2**—**2**, this initialization consists of the following code components: Open INTERNET Connection **234**—this part of the code opens an INTERNET Connection and sets the parameter for the connection. Then Set Callback Status routine **235** sets the callback status so as to inform the user of the status of connection. Finally Start New HTTP INTERNET Session **236** starts a new INTERNET session. The details of Communications Link **160** and the set up process **220**C are not critical, and will vary from platform to platform. Again, in some cases, users may use a low-speed dial-up connection, a dedicated high speed switched connection (T1 for example), an always-on xDSL connection, a wireless connection, and the like.

Iterative Processing of Queries/Answers

As illustrated in FIG. **3**, once initialization is complete, an iterative query/answer process is launched when the user presses the Start Button to initiate a query. Referring to FIG. **3**, the iterative query/answer process consists of two main sub-processes implemented as routines on the client side system **150**: Receive User Speech **240** and Receive User Answer **243**. The Receive User Speech **240** routine receives speech from the user (or another audio input source), while the Receive User Answer **243** routine receives an answer to the user's question in the form of text from the server so that it can be converted to speech for the user by text-to-speech engine **159**. As used herein, the term "query" is referred to in the broadest sense to refer, to either a question, a command, or some form of input used as a control variable by the system. For example, a query may consist of a question directed to a particular topic, such as "what is a network" in the context of a remote learning application. In an e-commerce application a query might consist of a command to "list all books by Mark Twain" for example. Similarly, while the answer in a remote learning application consists of text that is tendered into audible form by the text to speech engine **159**, it could also be returned as another form of multi-media information, such as a graphic image, a sound file, a video file, etc. depending on the requirements of the particular application. Again, given the present teachings concerning the necessary structure, operation, functions, performance, etc., of the client-side Receive User Speech **240** and Receiver User Answer **243** routines, one of ordinary skill in the art could implement such in a variety of ways.

Receive User Speech—As illustrated in FIG. **3**, the Receive User Speech routine **240** consists of a SRE **241** and a Communication **242** process, both implemented again as routines on the client side system **150** for receiving and partially processing the user's utterance. SRE routine **241** uses a coder **248** which is prepared so that a coder object receives speech data from a source object. Next the Start Source **249** routine is initiated. This part of the code initiates data retrieval using the source Object which will in turn be given to the Coder object. Next, MFCC vectors **250** are extracted from the Speech utterance continuously until silence is detected. As alluded to earlier, this represents the first phase of processing of the input speech signal, and in a preferred embodiment, it is intentionally restricted to merely computing the MFCC vectors for the reasons already expressed above. These vectors include the 12 cepstral coefficients and the RMS energy term, for a total of 13 separate numerical values for the partially processed speech signal.

In some environments, nonetheless, it is conceivable that the MFCC delta parameters and MFCC acceleration parameters can also be computed at client side system **150**, depending on the computation resources available, the transmission bandwidth in data link **160**A available to server side system **180**, the speed of a transceiver used for carrying data in the data link, etc. These parameters can be determined automatically by client side system upon initializing SRE **155** (using some type of calibration routine to measure resources), or by direct user control, so that the partitioning of signal processing responsibilities can be optimized on a case-by-case basis. In some applications, too, server side system **180** may lack the appropriate resources or routines for completing the processing of the speech input signal. Therefore, for some applications, the allocation of signal processing responsibilities may be partitioned differently, to the point where in fact both phases of the speech signal processing may take place at client side system **150** so that the speech signal is completely—rather than partially—processed and transmitted for conversion into a query at server side system **180**.

Again in a preferred embodiment, to ensure reasonable accuracy and real-time performance from a query/response perspective, sufficient resources are made available in a client side system so that 100 frames per second of speech data can be partially processed and transmitted through link

160A. Since the least amount of information that is necessary to complete the speech recognition process (only 13 coefficients) is sent, the system achieves a real-time performance that is believed to be highly optimized, because other latencies (i.e., client-side computational latencies, packet formation latencies, transmission latencies) are minimized. It will be apparent that the principles of the present invention can be extended to other SR applications where some other methodology is used for breaking down the speech input signal by an SRE (i.e., non-MFCC based). The only criteria is that the SR processing be similarly dividable into multiple phases, and with the responsibility for different phases being handled on opposite sides of link **160**A depending on overall system performance goals, requirements and the like. This functionality of the present invention can thus be achieved on a system-by-system basis, with an expected and typical amount of optimization being necessary for each particular implementation.

Thus, the present invention achieves a response rate performance that is tailored in accordance with the amount of information that is computed, coded and transmitted by the client side system **150**. So in applications where real-time performance is most critical, the least possible amount of extracted speech data is transmitted to reduce these latencies, and, in other applications, the amount of extracted speech data that is processed, coded and transmitted can be varied.

Communication—transmit communication module **242** is used to implement the transport of data from the client to the server over the data link **160**A, which in a preferred embodiment is the INTERNET. As explained above, the data consists of encoded MFCC vectors that will be used at then server-side of the Speech Recognition engine to complete the speech recognition decoding. The sequence of the communication is as follows:

OpenHTTPRequest **251**—this part of the code first converts MFCC vectors to a stream of bytes, and then processes the bytes so that it is compatible with a protocol known as HTTP. This protocol is well-known in the art, and it is apparent that for other data links another suitable protocol would be used.

1. Encode MFCC Byte Stream **251**—this part of the code encodes the MFCC vectors, so that they can be sent to the server via HTTP.
2. Send data **252**—this part of the code sends MFCC vectors to the server using the INTERNET connection and the HTTP protocol.

Wait for the Server Response **253**—this part of the code monitors the data link **160**A a response from server side system **180** arrives. In summary, the MFCC parameters are extracted or observed on-the-fly from the input speech signal. They are then encoded to a HTTP byte stream and sent in a streaming fashion to the server before the silence is detected—i.e. sent to server side system **180** before the utterance is complete. This aspect of the invention also facilitates a real-time behavior, since data can be transmitted and processed even while the user is still speaking.

Receive Answer from Server **243** is comprised of the following modules as shown in FIG. **3**.: MS Agent **244**, Text-to-Speech Engine **245** and receive communication modules **246**. All three modules interact to receive the answer from server side system **180**. As illustrated in FIG. **3**, the receive communication process consists of three separate processes implemented as a receive routine on client side system **150**: a Receive the Best Answer **258** receives the best answer over data link **160**B (the HTTP communication channel). The answer is de-compressed at **259** and then the answer is passed by code **260** to the MS Agent **244**, where it is received by code portion **254**. A routine **255** then articulates the answer using text-to-speech engine **257**. Of course, the text can also be displayed for additional feedback purposes on a monitor used with client side system **150**. The text to speech engine uses a natural language voice data file **256** associated with it that is appropriate for the particular language application (i.e., English, French, German, Japanese, etc.). As explained earlier when the answer is something more than text, it can be treated as desired to provide responsive information to the user, such as with a graphics image, a sound, a video clip, etc.

Uninitialization

The un-initialization routines and processes are illustrated in FIG. **4**. Three functional modules are used for un-initializing the primary components of the client side system **150**; these include SRE **270**, Communications **271** and MS Agent **272** un-initializing routines. To un-initialize SRE **220**A, memory that was allocated in the initialization phase is de-allocated by code **273** and objects created during such initialization phase are deleted by code **274**. Similarly, as illustrated in FIG. **4**, to un-initialize Communications module **220**C the INTERNET connection previously established with the server is closed by code portion **275** of the Communication Un-initialization routine **271**. Next the INTERNET session created at the time of initialization is also closed by routine **276**. For the un-initialization of the MS Agent **220**B, as illustrated in FIG. **4**, MS Agent Un-initialization routine **272** first releases the Commands Interface **227** using routine **277**. This releases the commands added to the property sheet during loading of the agent character by routine **225**. Next the Character Interface initialized by routine **226** is released by routine **278** and the Agent is unloaded at **279**. The Sink Object Interface is then also released **280** followed by the release of the Property Sheet Interface **281**. The Agent Notify Sink **282** then un-registers the Agent and finally the Agent Interface **283** is released which releases all the resources allocated during initialization steps identified in FIG. **2**—**2**.

It will be appreciated by those skilled in the art that the particular implementation for such un-initialization processes and routines in FIG. **4** will vary from client platform to client platform, as for the other routines discussed above. The structure, operation, etc. of such routines are well-known in the art, and they can be implemented using a number of fairly straightforward approaches without undue effort. Accordingly, they are not discussed in detail herein.

Description of Server Side System **180**

Introduction

A high level flow diagram of the set of preferred processes implemented on server side system **180** of Natural Language Query System **100** is illustrated in FIGS. **11**A through FIG. **11**C. In a preferred embodiment, this process consists of a two step algorithm for completing the processing of the speech input signal, recognizing the meaning of the user's query, and retrieving an appropriate answer/response for such query.

The 1st step as illustrated in FIG. **11**A can be considered a high-speed first-cut pruning mechanism, and includes the following operations: after completing processing of the speech input signal, the user's query is recognized at step **1101**, so that the text of the query is simultaneously sent to Natural Language Engine **190** (FIG. **1**) at step **1107**, and to DB Engine **186** (also FIG. **1**) at step **1102**. By "recognized"

in this context it is meant that the user's query is converted into a text string of distinct native language words through the HMM technique discussed earlier.

At NLE **190**, the text string undergoes morphological linguistic processing at step **1108**: the string is tokenized the tags are tagged and the tagged tokens are grouped Next the noun phrases (NP) of the string are stored at **1109**, and also copied and transferred for use by DB Engine **186** during a DB Process at step **1110**. As illustrated in FIG. **11**A, the string corresponding to the user's query which was sent to the DB Engine **186** at **1102**, is used together with the NP received from NLE **190** to construct an SQL Query at step **1103**. Next, the SQL query is executed at step **1104**, and a record set of potential questions corresponding to the user's query are received as a result of a full-text search at **1105**, which are then sent back to NLE **190** in the form of an array at step **1106**.

As can be seen from the above, this first step on the server side processing acts as an efficient and fast pruning mechanism so that the universe of potential "hits" corresponding to the user's actual query is narrowed down very quickly to a manageable set of likely candidates in a very short period of time.

Referring to FIG. **11**B, in contrast to the first step above, the 2^{nd} step can be considered as the more precise selection portion of the recognition process. It begins with linguistic processing of each of the stored questions in the array returned by the full-text search process as possible candidates representing the user's query. Processing of these stored questions continues in NLE **190** as follows: each question in the array of questions corresponding to the record set returned by the SQL full-text search undergoes morphological linguistic processing at step **1111**: in this operation, a text string corresponding to the retrieved candidate question is tokenized, the tags are tagged and the tagged tokens are grouped. Next, noun phrases of the string are computed and stored at step **1112**. This process continues iteratively at point **1113**, and the sequence of steps at **1118**, **1111**, **1112**, **1113** are repeated so that an NP for each retrieved candidate question is computed and stored. Once an NP is computed for each of the retrieved candidate questions of the array, a comparison is made between each such retrieved candidate question and the user's query based on the magnitude of the NP value at step **1114**. This process is also iterative in that steps **1114**, **1115**, **1116**, **1119** are repeated so that the comparison of the NP for each retrieved candidate question with that of the NP of the user's query is completed. When there are no more stored questions in the array to be processed at step **1117**, the stored question that has the maximum NP relative to the user's query, is identified at **1117**A as the stored question which best matches the user's query.

Notably, it can be seen that the second step of the recognition process is much more computationally intensive than the first step above, because several text strings are tokenized, and a comparison is made of several NPs. This would not be practical, nonetheless, if it were not for the fact that the first step has already quickly and efficiently reduced the candidates to be evaluated to a significant degree. Thus, this more computationally intensive aspect of the present invention is extremely valuable, however because it yields extremely high accuracy in the overall query recognition process. In this regard, therefore, this second step of the query recognition helps to ensure the overall accuracy of the system, while the first step helps to maintain a satisfactory speed that provides a real-time feel for the user.

As illustrated in FIG. **11**C, the last part of the query/response process occurs by providing an appropriate matching answer/response to the user. Thus, an identity of a matching stored question is completed at step **1120**. Next a file path corresponding to an answer of the identified matching question is extracted at step **1121**. Processing continues so that the answer is extracted from the file path at **1122** and finally the answer is compressed and sent to client side system **150** at step **1123**.

The discussion above is intended to convey a general overview of the primary components, operations, functions and characteristics of those portions of NLQS system **100** that reside on server side system **180**. The discussion that follows describes in more detail the respective sub-systems.

Software Modules used in Server Side System **180**

The key software modules used on server-side system **180** of the NLQS system are illustrated in FIG. **5**. These include generally the following components: a Communication module **500**—identified as CommunicationServer ISAPI **500**A (which is executed by SRE Server-side **182**—FIG. **1** and is explained in more detail below), and a database process DBProcess module **501** (executed by DB Engine **186**—FIG. **1**). Natural language engine module **500**C (executed by NLE **190**—FIG. **1**) and an interface **500**B between the NLE process module **500**C and the DBProcess module **500**B. As shown here, CommunicationServerISAPI **500**A includes a server-side speech recognition engine and appropriate communication interfaces required between client side system **150** and server side system **180**. As further illustrated in FIG. **5**, server-side logic of Natural Language Query System **100** also can be characterized as including two dynamic link library components: CommunicationServerISAPI **500** and DBProcess **501**. The CommunicationServerIASPI **500** is comprised of 3 sub-modules: Server-side Speech Recognition Engine module **500**A; Interface module **500**B between Natural Language Engine modules **500**C and DBProcess **501**; and the Natural Language Engine modules **500**C.

DB Process **501** is a module whose primary function is to connect to a SQL database and to execute an SQL query that is composed in response to the user's query. In addition, this module interfaces with logic that fetches the correct answer from a file path once this answer is passed to it from the Natural Language Engine module **500**C.

Speech Recognition Sub-System **182** on Server-Side System **180**

The server side speech recognition engine module **500**A is a set of distributed components that perform the necessary functions and operations of speech recognition engine **182** (FIG. **1**) at server-side **180**. These components can be implemented as software routines that are executed by server side **180** in conventional fashion. Referring to FIG. **4**A, a more detailed break out of the operation of the speech recognition components **600** at the server-side can be seen as follows:

Within a portion **601** of the server side SRE module **500**A, the binary MFCC vector byte stream corresponding to the speech signal's acoustic features extracted at client side system **150** and sent over the communication channel **160** is received. The MFCC acoustic vectors are decoded from the encoded HTTP byte stream as follows: Since the MFCC vectors contain embedded NULL characters, they cannot be transferred in this form to server side system **180** as such using HTTP protocol. Thus the MFCC vectors are first encoded at client-side **150** before transmission in such a way that all the speech data is converted into a stream of bytes

without embedded NULL characters in the data. At the very end of the byte stream a single NULL character is introduced to indicate the termination of the stream of bytes to be transferred to the server over the INTERNET **160**A using HTTP protocol.

As explained earlier, to conserve latency time between the client and server, a smaller number of bytes (just the 13 MFCC coefficients) are sent from client side system **150** to server side system **180**. This is done automatically for each platform to ensure uniformity, or can be tailored by the particular application environment—i.e., such as where it is determined that it will take less time to compute the delta and acceleration coefficients at the server (26 mote calculations), than it would take to encode them at the client, transmit them, and then decode them from the HTTP stream. In general, since server side system **180** is usually better equipped to calculate the MFCC delta and acceleration parameters, this is a preferable choice. Furthermore, there is generally more control over server resources compared to the client's resources, which means that future upgrades, optimizations, etc., can be disseminated and shared by all to make overall system performance more reliable and predictable. So, the present invention can accommodate even the worst-case scenario where the client's machine may be quite thin and may just have enough resources to capture the speech input data and do minimal processing.

Dictionary Preparation & Grammar Files

Referring to FIG. **4**A, within code block **605**, various options selected by the user (or gleaned from the user's status within a particular application) are received. For instance, in the case of a preferred remote learning system, Course, Chapter and/or Section data items are communicated. In the case of other applications (such as e-commerce) other data options are communicated, such as the Product Class, Product Category, Product Brand, etc. loaded for viewing within his/her browser. These selected options are based on the context experienced by the user during an interactive process, and thus help to limit and define the scope—i.e. grammars and dictionaries that will be dynamically loaded to speech recognition engine **182** (FIG. **1**) for Viterbi decoding during processing of the user speech utterance. For speech recognition to be optimized both grammar and dictionary files are used in a preferred embodiment. A Grammar file supplies the universe of available user queries; i.e., all the possible words that are to be recognized. The Dictionary file provides phonemes (the information of how a word is pronounced—this depends on the specific native language files that are installed—for example, UK English or US English) of each word contained in the grammar file. It is apparent that if all the sentences for a given environment that can be recognized were contained in a single grammar file then recognition accuracy would be deteriorated and the loading time alone for such grammar and dictionary files would impair the speed of the speech recognition process.

To avoid these problems, specific grammars are dynamically loaded or actively configured as the current grammar according to the user's context, i.e., as in the case of a remote learning system, the Course, Chapter and/or Section selected. Thus the grammar and dictionary files are loaded dynamically according to the given Course, Chapter and/or Section as dictated by the user, or as determined automatically by an application program executed by the user.

The second code block **602** implements the initialization of Speech Recognition engine **182** (FIG. **1**). The MFCC vectors received from client side system **150** along with the grammar filename and the dictionary file names are introduced to this block to initialize the speech decoder.

As illustrated in FIG. **4**A, the initialization process **602** uses the following sub-routines: A routine **602***a* for loading an SRE library. This then allows the creation of an object identified as External Source with code **602***b* using the received MFCC vectors. Code **602***c* allocates memory to hold the recognition objects. Routine **602***d* then also creates and initializes objects that are required for the recognition such as: Source, Coder, Recognizer and Results Loading of the Dictionary created by code **602***e*, Hidden Markov Models (HMMs) generated with code **602***f*; and Loading of the Grammar file generated by routine **602***g*.

Speech Recognition **603** is the next routine invoked as illustrated in FIG. **4**A, and is generally responsible for completing the processing of the user speech signals input on the client side **150**, which, as mentioned above, are preferably only partially processed (i.e., only MFCC vectors are computed during the first phase) when they are transmitted across link **160**. Using the functions created in External Source by subroutine **602***b*, this code reads MFCC vectors, one at a time from an External Source **603***a*, and processes them in block **603***b* to realize the words in the speech pattern that are symbolized by the MFCC vectors captured at the client. During this second phase, an additional 13 delta coefficients and an additional 13 acceleration coefficients are computed as part of the recognition process to obtain a total of 39 observation vectors O_t referred to earlier. Then, using a set of previously defined Hidden Markov Models (HMMs), the words corresponding to the user's speech utterance are determined in the manner described earlier. This completes the word "recognition" aspect of the query processing, which results are used further below to complete the query processing operations.

It will be appreciated by those skilled in the art that the distributed nature and rapid performance of the word recognition process, by itself, is extremely useful and may be implemented in connection with other environments that do not implicate or require additional query processing operations. For example, some applications may simply use individual recognized words for filling in data items on a computer generated form, and the aforementioned systems and processes can provide a rapid, reliable mechanism for doing so.

Once the user's speech is recognized, the flow of SRE **182** passes to Un-initialize SRE routine **604** where the speech engine is un-initialized as illustrated. In this block all the objects created in the initialization block are deleted by routine **604***a*, and memory allocated in the initialization block during the initialization phase are removed by routine **604***b*.

Again, it should be emphasized that the above are merely illustrative of embodiments for implementing the particular routines used on a server side speech recognition system of the present invention. Other variations of the same that achieve the desired functionality and objectives of the present invention will be apparent from the present teachings.

Database Processor **186** Operation—DBProcess

Construction of an SQL Query used as part of the user query processing is illustrated in FIG. **4**B, a SELECT SQL statement is preferably constructed using a conventional CONTAINS predicate. Module **950** constructs the SQL query based on this SELECT SQL statement, which query is used for retrieving the best suitable question stored in the database corresponding to the user's articulated query, (des-

ignated as Question here). A routine **951** then concatenates a table name with the constructed SELECT statement. Next, the number of words present in each Noun Phrase of Question asked by the user is calculated by routine **952**. Then memory is allocated by routine **953** as needed to accommodate all the words present in the NP. Next a word List (identifying all the distinct words present in the NP) is obtained by routine **954**. After this, this set of distinct words are concatenated by routine **955** to the SQL Query separated with a NEAR ( ) keyword. Next, the AND keyword is concatenated to the SQL Query by routine **956** after each NP. Finally memory resources are freed by code **957** so as to allocate memory to store the words received from NP for any next iteration. Thus, at the end of this process, a completed SQL Query corresponding to the user's articulated question is generated.

Connection to SQL Server—As illustrated in FIG. **4**C, after the SQL Query is constructed by routine **710**, a routine **711** implements a connection to the query database **717** to continue processing of the user query. The connection sequence and the subsequent retrieved record set is implemented using routines **700** which include the following:

1. Server and database names are assigned by routine **711**A to a DBProcess member variable
2. A connection string is established by routine **711**B;
3. The SQL Server database is connected under control of code **711**C
4. The SQL Query is received by routine **712**A
5. The SQL Query is executed by code **712**B
6. Extract the total number of records retrieved by the query—**713**
7. Allocate the memory to store the total number of paired questions—**713**
8. Store the entire number of paired questions into an array—**713**

Once the Best Answer ID is received at **716** FIG. **4**C, from the NLE **14** (FIG. **5**), the code corresponding **716**C receives it passes it to code in **716**B where the path of the Answer file is determined using the record number. Then the file is opened **716**C using the path passed to it and the contents of the file corresponding to the answer is read. Then the answer is compressed by code in **716**D and prepared for transmission over the communication channel **160**B (FIG. **1**).

NLQS Database **188**—Table Organization

FIG. **6** illustrates a preferred embodiment of a logical structure of tables used in a typical NLQS database **188** (FIG. **1**). When NLQS database **188** is used as part of NLQS query system **100** implemented as a remote learning/training environment, this database will include an organizational multi-level hierarchy that consists typically of a Course **701**, which is made of several chapters **702**, **703**, **704**. Each of these chapters can have one or more Sections **705**, **706**, **707** as shown for Chapter 1. A similar structure can exist for Chapter 2, Chapter 3 . . . Chapter N. Each section has a set of one or more question—answer pairs **708** stored in tables described in more detail below. While this is an appropriate and preferable arrangement for a training/learning application, it is apparent that other implementations would be possible and perhaps more suitable for other applications such as e-commerce, e-support, INTERNET browsing, etc., depending on overall system parameters.

It can be seen that the NLQS database **188** organization is intricately linked to the switched grammar architecture described earlier. In other words, the context (or environment) experienced by the user can be determined at any moment in time based at the selection made at the section level, so that only a limited subset of question-answer pairs **708** for example are appropriate for section **705**. This in turn means that only a particular appropriate grammar for such question-answer pairs may be switched in for handling user queries while the user is experiencing such context. In a similar fashion, an e-commerce application for an INTERNET based business may consist of a hierarchy that includes a first level "home" page **701** identifying user selectable options (product types, services, contact information, etc.), a second level may include one or more "product types" pages **702**, **703**, **704**, a third page may include particular product models **705**, **706**, **707**, etc., and with appropriate question-answer pairs **708** and grammars customized for handling queries for such product models. Again, the particular implementation will vary from application to application, depending on the needs and desires of such business, and a typical amount of routine optimization will be necessary for each such application.

Table Organization

In a preferred embodiment, an independent database is used for each Course. Each database in turn can include three types of tables as follows: a Master Table as illustrated in FIG. **7**A, at least one Chapter Table as illustrated in FIG. **7**B and at least one Section Table as illustrated in FIG. **7**C.

As illustrated in FIG. **7**A, a preferred embodiment of a Master Table has six columns—Field Name **701**A, Data Type **702**A, Size **703**A, Null **704**A, Primary Key **705**A and Indexed **706**A. These parameters are well-known in the art of database design and structure. The Master Table has only two fields—Chapter Name **707**A and Section Name **708**A. Both ChapterName and Section Name are commonly indexed.

A preferred embodiment of a Chapter Table is illustrated in FIG. **7**B. As with the Master Table, the Chapter Table has six (6) columns—Field Name **720**, Data Type **721**, Size **722**, Null **723**, Primary Key **724** and Indexed **725**. There are nine (9) rows of data however, in this case,—Chapter_ID **726**, Answer_ID **727**, Section Name **728**, Answer_Title **729**, PairedQuestion **730**, AnswerPath **731**, Creator **732**, Date of Creation **733** and Date of Modification **734**.

An explanation of the Chapter Table fields is provided in FIG. **7**C. Each of the eight (8) Fields **720** has a description **735** and stores data corresponding to:

- AnswerID **727**—an integer that is automatically incremented for each answer given for user convenience
- Section_Name **728**—the name of the section to which the particular record belongs. This field along with the AnswerID is used as the primary key
- Answer_Title **729**—A short description of the title of the answer to the user query
- PairedQuestion **730**—Contains one or more combinations of questions for the related answers whose path is stored in the next column AnswerPath
- AnswerPath **731**—contains the path of a file, which contains the answer to the related questions stored in the previous column; in the case of a pure question/answer application, this file is a text file, but, as mentioned above, could be a multi-media file of any kind transportable over the data link **160**
- Creator **732**—Name of Content Creator
- Date_of_Creation **733**—Date on which content was created
- Date of Modification **734**—Date on which content was changed or modified

A preferred embodiment of a Section Table is illustrated in FIG. **7**D. The Section Table has six (6) columns—Field

Name **740**, Data Type **741**, Size **742**, Null **743**, Primary Key **744** and Indexed **745**. There are seven (7) rows of data—Answer_ID **746**, Answer_Title **747**, PairedQuestion **748**, AnswerPath **749**, Creator **750**, Date of Creation **751** and Date of Modification **752**. These names correspond to the same fields, columns already described above for the Master Table and Chapter Table.

Again, this is a preferred approach for the specific type of learning/training application described herein. Since the number of potential applications for the present invention is quite large, and each application can be customized, it is expected that other applications (including other learning/training applications) will require and/or be better accommodated by another table, column, and field structure/hierarchy.

Search Service and Search Engine—A query text search service is performed by an SQL Search System **1000** shown in FIG. **10**. This system provides querying support to process full-text searches. This is where full-text indexes reside.

In general, SQL Search System determines which entries in a database index meet selection criteria specified by a particular text query that is constructed in accordance with an articulated user speech utterance. The Index Engine **1011**B is the entity that populates the Full-Text Index tables with indexes which correspond to the indexable units of text for the stored questions and corresponding answers. It scans through character strings, determines word boundaries, removes all noise words and then populates the full-text index with the remaining words. For each entry in the full text database that meets the selection criteria, a unique key column value and a ranking value are returned as well. Catalog set **1013** is a file-system directory that is accessible only by an Administrator and Search Service **1010**. Full-text indexes **1014** are organized into full-text catalogs, which are referenced by easy to handle names. Typically, full-text index data for an entire database is placed into a single full-text catalog.

The schema for the full-text database as described (FIG. **7**, FIG. **7**A, FIG. **7**B, FIG. **7**C, FIG. **7**D) is stored in the tables **1006** shown in FIG. **10**. Take for example, the tables required to describe the structure the stored question/answer pairs required for a particular course. For each table—Course Table, Chapter Table, Section Table, there are fields—column information that define each parameters that make up the logical structure of the table. This information is stored in User and System tables **1006**. The key values corresponding to those tables are stored as Full-Text catalogs **1013**. So when processing a full-text search, the search engine returns to the SQL Server the key values of the rows that match the search criteria. The relational engine then uses this information to respond to the query.

As illustrated in FIG. **10**, a Full-Text Query Process is implemented as follows:

1. A query **1001** that uses a SQL full-text construct generated by DB processor **186** is submitted to SQL Relational Engine **1002**.
2. Queries containing either a CONTAINS or FREETEXT predicate are rewritten by routine **1003** so that a responsive rowset returned later from Full-Text Provider **1007** will be automatically joined to the table that the predicate is acting upon. This rewrite is a mechanism used to ensure that these predicates are a seamless extension to a traditional SQL Server. After the compiled query is internally rewritten and checked for correctness in item **1003**, the query is passed to RUN TIME module **1004**. The function of module **1004** is to convert the rewritten SQL construct to a validated run-time process before it is sent to the Full-Text Provider, **1007**.
3. After this, Full-Text Provider **1007** is invoked, passing the following information for the query:
   a. A ft_search_condition parameter (this is a logical flag indicating a full text search condition)
   b. A name of a full-text catalog where a full-text index of a table resides
   c. A locale ID to be used for language (for example, word breaking)
   d. Identities of a database, table, and column to be used in the query
   e. If the query is comprised of more than one full-text construct; when this is the case Full-text provider **1007** is invoked separately for each construct.
4. SQL Relational Engine **1002** does not examine the contents of ft_search_condition. Instead, this information is passed along to Full-text provider **1007**, which verifies the validity of the query and then creates an appropriate internal representation of the full-text search condition.
5. The query request/command **1008** is then passed to Querying Support **1011**A.
6. Querying Support **1012** returns a rowset **1009** from Full-Text Catalog **1013** that contains unique key column values for any rows that match the full-text search criteria. A rank value also is returned for each row.
7. The rowset of key column values **1009** is passed to SQL Relational Engine **1002**. If processing of the query implicates either a CONTAINSTABLE( ) or FREETEXTTABLE( ) function, RANK values are returned; otherwise, any rank value is filtered out.
8. The rowset values **1009** are plugged into the initial query with values obtained from relational database **1006**, and a result set **1015** is then returned for further processing to yield a response to the user.

At this stage of the query recognition process, the speech utterance by the user has already been rapidly converted into a carefully crafted text query, and this text query has been initially processed so that an initial matching set of results can be further evaluated for a final determination of the appropriate matching question/answer pair. The underlying principle that makes this possible is the presence of a full-text unique key column for each table that is registered for full-text searches. Thus when processing a full-text search, SQL Search Service **1010** returns to SQL server **1002** the key values of the rows that match the database. In maintaining these full-text databases **1013** and full text indexes **1014**, the present invention has the unique characteristic that the full-text indices **1014** are not updated instantly when the full-text registered columns are updated. This operation is eliminated, again, to reduce recognition latency, increase response speed, etc. Thus, as compared to other database architectures, this updating of the full-text index tables, which would otherwise take a significant time, is instead done asynchronously at a more convenient time.

Interface Between NLE **190** and DB Processor **188**

The result set **1015** of candidate questions corresponding to the user query utterance are presented to NLE **190** for further processing as shown in FIG. **4**D to determine a "best" matching question/answer pair. An NLE/DBProcessor interface module coordinates the handling of user queries, analysis of noun-phrases (NPs) of retrieved questions sets from the SQL query based on the user query, comparing the retrieved question NPs with the user query NP, etc. between NLE **190** and DB Processor **188**. So, this part of the

server side code contains functions, which interface processes resident in both NLE block **190** and DB Processor block **188**. The functions are illustrated in FIG. **4**D; As seen here, code routine **880** implements functions to extract the Noun Phrase (NP) list from the user's question. This part of the code interacts with NLE **190** and gets the list of Noun Phrases in a sentence articulated by the user. Similarly, Routine **813** retrieves an NP list from the list of corresponding candidate/paired questions **1015** and stores these questions into an (ranked by NP value) array. Thus, at this point, NP data has been generated for the user query, as well as for the candidate questions **1015**. As an example of determining the noun phrases of a sentence such as: "What issues have guided the President in considering the impact of foreign trade policy on American businesses?" NLE **190** would return the following as noun phrases: President, issues, impact of foreign trade policy, American businesses, impact, impact of foreign trade, foreign trade, foreign trade policy, trade, trade policy, policy, businesses. The methodology used by NLE **190** will thus be apparent to those skilled in the art from this set of noun phrases and noun sub-phrases generated in response to the example query.

Next, a function identified as Get Best Answer ID **815** is implemented. This part of the code gets a best answer ID corresponding to the user's query. To do this, routines **813**A, **813**B first find out the number of Noun phrases for each entry in the retrieved set **1015** that match with the Noun phrases in the user's query. Then routine **815***a* selects a final result record from the candidate retrieved set **1015** that contains the maximum number of matching Noun phrases.

Conventionally, nouns are commonly thought of as "naming" words, and specifically as the names of "people, places, or things". Nouns such as John, London, and computer certainly fit this description, but the types of words classified by the present invention as nouns is much broader than this. Nouns can also denote abstract and intangible concepts such as birth, happiness, evolution, technology, management, imagination, revenge, politics, hope, cooker, sport, and literacy. Because of the enormous diversity of nouns compared to other parts of speech, the Applicant has found that it is much more relevant to consider the noun phrase as a key linguistic metric. So, the great variety of items classified as nouns by the present invention helps to discriminate and identify individual speech utterances much easier and faster than prior techniques disclosed in the art.

Following this same thought, the present invention also adopts and implements another linguistic entity—the word phrase—to facilitate speech query recognition. The basic structure of a word phrase—whether it be a noun phrase, verb phrase, adjective phrase—is three parts—[pre-Head string], [Head] and [post-Head string]. For example, in the minimal noun phrase—"the children," "children" is classified as the Head of the noun phrase. In summary, because of the diversity and frequency of noun phrases, the choice of noun phrase as the metric by which stored answer is linguistically chosen, has a solid justification in applying this technique to the English natural language as well as other natural languages. So, in sum, the total noun phrases in a speech utterance taken together operate extremely well as unique type of speech query fingerprint.

The ID corresponding to the best answer corresponding to the selected final result record question is then generated by routine **815** which then returns it to DB Process shown in FIG. **4**C. As seen there, a Best Answer ID I is received by routine **716**A, and used by a routine **716**B to retrieve an answer file path. Routine **716**C then opens and reads the answer file, and communicates the substance of the same to routine **716**D. The latter then compresses the answer file data, and sends it over data link **160** to client side system **150** for processing as noted earlier (i.e., to be rendered into audible feedback, visual text/graphics, etc.). Again, in the context of a learning/instructional application, the answer file may consist solely of a single text phrase, but in other applications the substance and format will be tailored to a specific question in an appropriate fashion. For instance, an "answer" may consist of a list of multiple entries corresponding to a list of responsive category items (i.e., a list of books to a particular author) etc. Other variations will be apparent depending on the particular environment.

Natural Language Engine **190**

Again referring to FIG. **4**D, the general structure of NL engine **190** is depicted. This engine implements the word analysis or morphological analysis of words that make up the user's query, as well as phrase analysis of phrases extracted from the query.

As illustrated in FIG. **9**, the functions used in a morphological analysis include tokenizers **802**A, stemmers **804**A and morphological analyzers **806**A. The functions that comprise the phrase analysis include tokenizers, taggers and groupers, and their relationship is shown in FIG. **8**.

Tokenizer **802**A is a software module that functions to break up text of an input sentence **801**A into a list of tokens **803**A. In performing this function, tokenizer **802**A goes through input text **801**A and treats it as a series of tokens or useful meaningful units that are typically larger than individual characters, but smaller than phrases and sentences. These tokens **803**A can include words, separable parts of word and punctuation. Each token **803**A is given an offset and a length. The first phase of tokenization is segmentation, which extracts the individual tokens from the input text and keeps track of the offset where each token originated from in the input text. Next, categories are associated with each token, based on its shape. The process of tokenization is well-known in the art, so it can be performed by any convenient application suitable for the present invention.

Following tokenization, a stemmer process **804**A is executed, which can include two separate forms—inflectional and derivational, for analyzing the tokens to determine their respective stems **805**A. An inflectional stemmer recognizes affixes and returns the word which is the stem. A derivational stemmer on the other hand recognizes derivational affixes and returns the toot word or words. While stemmer **804**A associates an input word with its stem, it does not have parts of speech information. Analyzer **806**B takes a word independent of context, and returns a set of possible parts of speech **806**A.

As illustrated in FIG. **8**, phrase analysis **800** is the next step that is performed after tokenization. A tokenizer **802** generates tokens from input text **801**. Tokens **803** are assigned to parts of a speech tag by a tagger routine **804**, and a grouper routine **806** recognizes groups of words as phrases of a certain syntactic type. These syntactic types include for example the noun phrases mentioned earlier, but could include other types if desired such as verb phrases and adjective phrases. Specifically, tagger **804** is a parts-of-speech disambiguator, which analyzes words in context. It has a built-in morphological analyzer (not shown) that allows it to identify all possible parts of speech for each token. The output of tagger **804** is a string with each token tagged with a parts-of-speech label **805**. The final step in the linguistic process **800** is the grouping of words to form phrases **807**. This function is performed by the grouper **806**,

and is very dependent, of course, on the performance and output of tagger component **804**.

Accordingly, at the end of linguistic processing **800**, a list of noun phrases (NP) **807** is generated in accordance with the user's query utterance. This set of NPs generated by NLE **190** helps significantly to refine the search for the best answer, so that a single-best answer can be later provided for the user's question.

The particular components of NLE **190** are shown in FIG. **4**D, and include several components. Each of these components implement the several different functions required in NLE **190** as now explained.

Initialize Grouper Resources Object and the Library **900**—this routine initializes the structure variables required to create grouper resource object and library. Specifically, it initializes a particular natural language used by NLE **190** to create a Noun Phrase, for example the English natural language is initialized for a system that serves the English language market. In turn, it also creates the objects (routines) required for Tokenizer, Tagger and Grouper (discussed above) with routines **900**A, **900**B, **900**C and **900**D respectively, and initializes these objects with appropriate values. It also allocates memory to store all the recognized Noun Phrases for the retrieved question pairs.

Tokenizing of the words from the given text (from the query or the paired questions) is performed with routine **909**B—here all the words are tokenized with the help of a local dictionary used by NLE **190** resources. The resultant tokenized words are passed to a Tagger routine **909**C. At routine **909**C, tagging of all the tokens is done and the output is passed to a Grouper routine **909**D.

The Grouping of all tagged token to form NP list is implemented by routine **909**D so that the Grouper groups all the tagged token words and outputs the Noun Phrases.

Un-initializing of the grouper resources object and freeing of the resources, is performed by routines **909**EA, **909**EB and **909**EC. These include Token Resources, Tagger Resources and Grouper Resources respectively. After initialization, the resources are freed. The memory that was used to store all Noun Phrases are also de-allocated.

Additional Embodiments

In a e-commerce embodiment of the present invention as illustrated in FIG. **13**, a web page **1300** contains typical visible links such as Books **1310**, Music **1320** so that on clicking the appropriate link the customer is taken to those pages. The web page may be implemented using HTML, a Java applet, or similar coding techniques which interact with the user's browser. For example, if customer wants to buy an album C by Artist Albert, he traverses several web pages as follows: he first clicks on Music (FIG. **13**, **1360**), which brings up page **1400** where he/she then clicks on Records (FIG. **14**, **1450**). Alternatively, he/she could select CDs **1460**, Videos **1470**, or other categories of books **1410**, music **1420** or help **1430**. As illustrated in FIG. **15**, this brings up another web page **1500** with links for Records **1550**, with sub-categories—Artist **1560**, Song **1570**, Title **1580**, Genre **1590**. The customer must then click on Artist **1560** to select the artist of choice. This displays another web page **1600** as illustrated in FIG. **16**. On this page the various artists **1650** are listed as illustrated—Albert **1650**, Brooks **1660**, Charlie **1670**, Whyte **1690** are listed under the category Artists **1650**. The customer must now click on Albert **1660** to view the albums available for Albert. When this is done, another web page is displayed as shown in FIG. **17**. Again this web page **1700** displays a similar look and feel, but with the albums available **1760**, **1770**, **1780** listed under the heading Tides **1750**. The customer can also read additional information **1790** for each album. This album information is similar to the liner notes of a shrink-wrapped album purchased at a retail store. One Album A is identified, the customer must click on the Album A **1760**. This typically brings up another text box with the information about its availability, price, shipping and handling charges etc.

When web page **1300** is provided with functionality of a NLQS of the type described above, the web page interacts with the client side and server side speech recognition modules described above. In this case, the user initiates an inquiry by simply clicking on a button designated Contact Me for Help **1480** (this can be a link button on the screen, or a key on the keyboard for example) and is then told by character **1440** about how to elicit the information required. If the user wants Album A by artist Albert, the user could articulate "Is Album A by Brooks available?" in much the same way they would ask the question of a human clerk at a brick and mortar facility. Because of the rapid recognition performance of the present invention, the user's query would be answered in real-time by character **1440** speaking out the answer in the user's native language. If desired, a readable word balloon **1490** could also be displayed to see the character's answer and so that save/print options can also be implemented. Similar appropriate question/answer pairs for each page of the website can be constructed in accordance with the present teachings, so that the customer is provided with an environment that emulates a normal conversational human-like question and answer dialog for all aspects of the web site. Character **1440** can be adjusted and tailored according to the particular commercial application, or by the user's own preferences, etc. to have a particular voice style (man, woman, young, old, etc.) to enhance the customer's experience.

In a similar fashion, an articulated user query might be received as part of a conventional search engine query, to locate information of interest on the INTERNET in a similar manner as done with conventional text queries. If a reasonably close question/answer pair is not available at the server side (for instance, if it does not reach a certain confidence level as an appropriate match to the user's question) the user could be presented with the option of increasing the scope so that the query would then be presented simultaneously to one or more different NLEs across a number of servers, to improve the likelihood of finding an appropriate matching question/answer pair. Furthermore, if desired, more than one "match" could be found, in the same fashion that conventional search engines can return a number of potential "hits" corresponding to the user's query. For some such queries, of course, it is likely that real-time performance will not be possible (because of the disseminated and distributed processing) but the advantage presented by extensive supplemental question/answer database systems may be desirable for some users.

It is apparent as well that the NLQS of the present invention is very natural and saves much time for the user and the e-commerce operator as well. In an e-support embodiment, the customer can retrieve information quickly and efficiently, and without need for a live customer agent. For example, at a consumer computer system vendor related support site, a simple diagnostic page might be presented for the user, along with a visible support character to assist him/her. The user could then select items from a "symptoms" page (i.e., a "monitor" problem, a "keyboard" problem, a "printer" problem, etc.) simply by articulating such symptoms in response to prompting from the support character. Thereafter, the system will direct the user on a

real-time basis to more specific sub-menus, potential solutions, etc. for the particular recognized complaint. The use of a programmable character thus allows the web site to be scaled to accommodate a large number of hits or customers without any corresponding need to increase the number of human resources and its attendant training issues.

As an additional embodiment, the searching for information on a particular web site may be accelerated with the use of the NLQS of the present invention. Additionally, a significant benefit is that the information is provided in a user-friendly manner through the natural interface of speech. The majority of web sites presently employ lists of frequently asked questions which the user typically wades item by item in order to obtain an answer to a question or issue. For example, as displayed in FIG. **13**, the customer clicks on Help **1330** to initiate the interface with a set of lists. Other options include computer related items at **1370** and frequently asked questions (FAQ) at **1380**.

As illustrated in FIG. **18**, a web site plan for typical web page is displayed. This illustrates the number of pages that have to be traversed in order to reach the list of Frequently-Asked Questions. Once at this page, the user has to scroll and manually identify the question that matches his/her query. This process is typically a laborious task and may or may not yield the information that answers the user's query. The present art for displaying this information is illustrated in FIG. **18**. This figure identifies how the information on a typical web site is organized: the Help link (FIG. **13**, **1330**) typically shown on the home page of the web page is illustrated shown on FIG. **18** as **1800**. Again referring to FIG. **18**, each sub-category of information is listed on a separate page. For example, **1810** lists sub-topics such as 'First Time Visitors', 'Search Tips', 'Ordering', 'Shipping', 'Tour Account' etc. Other pages deal with 'Account information' **1860**, 'Rates and Policies' **1850** etc. Down another level, there are pages that deal exclusively with a sub—sub topics on a specific page such as 'First Time Visitors' **1960**, 'Frequently Asked Questions' **1950**, 'Safe Shopping Guarantee' **1940**, etc. So if a customer has a query that is best answered by going to the Frequently Asked Questions link, he or she has to traverse three levels of busy and cluttered screen pages to get to the Frequently Asked Questions page **1950**. Typically, there are many lists of questions **1980** that have to be manually scrolled through. While scrolling visually, the customer then has to visually and mentally match his or her question with each listed question. If a possible match is sighted, then that question is clicked and the answer then appears in text form which then is read.

In contrast, the process of obtaining an answer to a question using a web page enabled with the present NLQS can be achieved much less laboriously and efficiently. The user would articulate the word "Help" (FIG. **13**, **1330**). This would immediately cause a character (FIG. **13**, **1340**) to appear with the friendly response "May I be of assistance. Please state your question?". Once the customer states the question, the character would then perform an animation or reply "Thank you, I will be back with the answer soon". After a short period time (preferably not exceeding 5–7 seconds) the character would then speak out the answer to the user's question. As illustrated in FIG. **18** the answer would be the answer **1990** returned to the user in the form of speech is the answer that is paired with the question **1950**. For example, the answer **1990**: "We accept Visa, MasterCard and Discover credit cards", would be the response to the query **2000** "What forms of payments do you accept?"

Another embodiment of the invention is illustrated in FIG. **12**. This web page illustrates a typical website that employs NLQS in a web-based learning environment. As illustrated in FIG. **12**, the web page in browser **1200**, is divided into two or more frames. A character **1210** in the likeness of an instructor is available on the screen and appears when the student initiates the query mode either by speaking the word "Help" into a microphone (FIG. **2**, **215**) or by clicking on the link 'Click to Speak' (FIG. **12**, **1280**). Character **1210** would then prompt the student to select a course **1220** from the drop down list **1230**. If the user selects the course 'CPlusPlus', the character would then confirm verbally that the course "CPlusPlus" was selected. The character would then direct the student to make the next selection from the drop-down list **1250** that contains the selections for the chapters **1240** from which questions are available. Again, after the student makes the selection, the character **1210** confirms the selection by speaking. Next character **1210** prompts the student to select 'Section' **1260** of the chapter from which questions are available from the drop down list **1270**. Again, after the student makes the selection, character **1210** confirms the selection by articulating the 'Section' **1260** chosen. As a prompt to the student, a list of possible questions appear in the list box **1291**. In addition, tips **1290** for using the system are displayed. Once the selections are all made, the student is prompted by the character to ask the question as follows: "Please ask your query now". The student then speaks his query and after a short period of time, the character responds with the answer preceded by the question as follows: "The answer to your question . . . is as follows: . . . ". This procedure allows the student to quickly retrieve answers to questions about any section of the course and replaces the tedium of consulting books, and references or indices. In short, it is can serve a number of uses from being a virtual teacher answering questions on-the-fly or a flash card substitute.

From preliminary data available to the inventors, it is estimate that the system can easily accommodate 100–250 question/answer pairs while still achieving a real-time feel and appearance to the user (i.e., less than 10 seconds of latency, not counting transmission) using the above described structures and methods. It is expected, of course, that these figures will improve as additional processing speed becomes available, and routine optimizations are employed to the various components noted for each particular environment.

Again, the above are merely illustrative of the many possible applications of the present invention, and it is expected that many more web-based enterprises, as well as other consumer applications (such as intelligent, interactive toys) can utilize the present teachings. Although the present invention has been described in terms of a preferred embodiment, it will be apparent to those skilled in the art that many alterations and modifications may be made to such embodiments without departing from the teachings of the present invention. It will also be apparent to those skilled in the art that many aspects of the present discussion have been simplified to give appropriate weight and focus to the more germane aspects of the present invention. The microcode and software routines executed to effectuate the inventive methods may be embodied in various forms, including in a permanent magnetic media, a non-volatile ROM, a CD-ROM, or any other suitable machine-readable format. Accordingly, it is intended that the all such alterations and modifications be included within the scope and spirit of the invention as defined by the following claims.

What is claimed is:

1. A speech-enabled internet website operating on a server computing system and comprising:

a receiving routine executing on the server computing system for receiving speech data associated with a user speech-based query, said speech data being characterized by a data content that is substantially inadequate by itself for permitting recognition of words articulated in said speech query; and

a speech recognition routine executing on the server computing system for completing recognition of said speech query using said speech data and said data content to generate a recognized speech query; and

a web page having a list of items, at least some of said list of items being selectable by a user based on said recognized speech query;

wherein signal processing functions required to generate said recognized speech query can be allocated between a client platform and the server computing system as needed based on computing resources available to said client platform and server computing system respectively.

2. The website of claim **1**, wherein said web page displays an additional list of one or more items based on said recognized speech query.

3. The website of claim **1**, wherein said website is adapted so that the user can navigate and locate information of interest using said speech query.

4. The website of claim **1**, wherein said list of items include products and/or services offered by said website.

5. The website of claim **1**, wherein said web page is implemented in HTML or as a Java applet.

6. The website of claim **1**, wherein said website is further adapted to respond to a speech query concerning said list of items by returning a text or speech articulated response.

7. The website of claim **1**, wherein said website is further adapted to interact on a real-time basis in response to one or more continuous speech queries.

8. The website of claim **1**, wherein said speech recognition routine can complete recognition of said speech query with less latency than would that resulting if said additional data content were generated by a client platform used by the user.

9. The website of claim **1**, wherein said data content constitutes a minimum amount of information that can be used by said speech recognition engine to complete accurate recognition of words and sentences in said speech query.

10. The website of claim **1**, wherein the website also controls an interactive character agent presented to the user for assisting in handling said speech query.

11. The system of claim **10**, wherein said interactive character agent provides suggestions for queries which the user can articulate.

12. The system of claim **10**, wherein a different interactive character agent can be presented to different users providing speech utterances received by the server computing system.

13. The system of claim **10**, wherein said interactive character agent is configured to perform a dialog of successive questions and answers with the user during an interactive session.

14. The system of claim **10**, wherein said server computing system causes said interactive character agent to respond in real-time whenever the user provides selected speech input.

15. The website of claim **1**, wherein said list of items correspond to topics associated with an interactive lesson tutorial.

16. The system of claim **1** wherein respective signal processing functions to be performed by the client platform and the server computing system are specified by an initialization routine.

17. The system of claim **16** wherein respective signal processing functions to be performed by the client platform and the server computing system are further specified in accordance with transmission characteristics associated with a communications channel used for said speech data.

18. The system of claim **1** wherein the server computing system is adapted to handle a client platform that can include a plurality of different hand held computing devices covering a range of differing respective computing capabilities.

19. The system of claim **1**, wherein said speech data is formatted by a client device with at least one predetermined character used to designate end of an utterance.

20. The system of claim **19**, wherein said predetermined character is a NULL character.

21. The system of claim **1**, wherein the server computing system transfers speech related data for the web page using a hypertext transfer protocol (HTTP).

22. The system of claim **1**, wherein a signal processing function performed by the client platform includes generating at least partial speech observation vectors using mel frequency cepstral coefficients.

23. The system of claim **1**, wherein a signal processing function performed by the client platform includes at least calibrating speech and silence components of a speech utterance.

24. The system of claim **1** wherein the server computing system is further configured to perform a natural language processing operation on said recognized speech query to recognize a meaning of a sentence of words contained therein.

25. The system of claim **24** wherein said server computing system includes a plurality of separate natural language engines.

26. The system of claim **24** wherein said natural language processing operation is configured to compare a limited set of phrases from said recognized speech query with a separate set of phrases corresponding to predefined valid queries from users.

27. The system of claim **24** wherein text from said recognized speech query is presented to both a natural language engine for performing said natural language processing operation as well as to a database for identifying a meaning of said recognized speech query, such that a response can be provided by said database for at least some recognized speech queries before said natural language processing operation is completed.

28. The system of claim **24**, further including a database query engine which performs part of said natural language operation by combining said speech query with search predicates to retrieve from a database a set of one or more potential responsive answers to said speech query.

29. The system of claim **1** wherein the server computing system is further configured to dynamically change a speech recognition grammar based on input provided by a user to selections available within said web page.

30. The system of claim **29** wherein multiple speech grammars are available and selectable within the web page, and such that speech input provided by the user for an item within the web page using a first grammar dynamically controls which one of a plurality of second grammars is loaded for speech recognition of subsequent speech input by the user.

31. The system of claim 29 wherein multiple speech grammars are selectable in a hierarchy within the web page, such that speech input provided by the user for an item within a first level menu of the web page using a first grammar dynamically controls which one of a plurality of second grammars at a second level menu of the web page and/or a second web page is loaded for speech recognition.

32. The system of claim 1 wherein the server computing system is further configured to dynamically change a speech recognition grammar based on spoken responses provided by a user during a real-time dialogue session conducted with an interactive electronic agent associated with said web page.

33. The system of claim 1, wherein said web page includes first tags which are selectable by one of a pointing device or a keyboard, and separate second tags selectable by speech input.

34. The system of claim 1, wherein said web page includes tags which can be selected by a pointing device and/or a keyboard and/or speech input.

35. The system of claim 1, wherein said web page and associated speech data is communicated to a client device using a hypertext transfer protocol (HTTP).

36. The system of claim 1, wherein said server computing system includes text to speech capability for outputting a response associated with said web page in audible form.

37. The system of claim 1, wherein said speech query is recognized by forming a concatenation of words and/or phrases derived from said speech query and using said concatenation as a search query for a database.

38. The system of claim 1, wherein the user can speak a help command while interacting with any web page maintained by the server computing system to cause an interactive character agent to appear.

39. A speech-enabled internet website operating on a server computing system and comprising:

a receiving routine executing on the server computing system for receiving speech data associated with a user speech-based query, said speech data being characterized by a first data content that is substantially inadequate by itself for permitting recognition of words articulated in said speech query; and

a speech recognition routine executing on the server computing system for completing recognition of said speech query using said speech data and said first data content to generate a recognized speech query; and

a web page having a search engine for locating user selected information of interest, said search engine using a text query that is derived from said recognized speech query;

wherein signal processing functions required to generate said recognized speech query can be allocated between a client platform and the server computing system as needed based on computing resources available to said client platform and server computing system respectively.

40. The website of claim 39, wherein said speech query is processed by more than one server computing system, so that multiple search engines are used for locating said information of interest.

41. The website of claim 39, wherein said web page includes a list of one or more items associated with assisting a user to diagnose a product or service problem, and which one or more items are also selectable by a user speech-based query.

42. The website of claim 39, wherein said website provides and controls an agent for assisting a user to interact with said website.

43. The website of claim 39, wherein said list of items correspond to topics associated with an interactive lesson tutorial.

44. A system for enabling a user web browser program to interact with a website using speech utterances, the system comprising:

a receiving routine for receiving speech data associated with a speech utterance generated at a client platform, said speech data being characterized by a limited speech data content to reduce processing and transmission latencies; and

a speech recognition routine executing on a server computing system for completing recognition of said speech utterance using said limited speech data content to generate a recognized speech query in real-time; and

a web page routine for presenting one or more web pages to the user web browser program, wherein data content for said one or more web pages perceived by the user is controlled by said recognized speech query;

wherein signal processing functions required to generate said recognized speech query can be allocated between a client platform and the server computing system as needed based on computing resources available to said client platform and server computing system respectively.

45. The system of claim 44, wherein said recognized speech query can include one of a number of predefined sentences recognizable by said system, and said speech query is recognized by identifying a candidate set of potential sentences from a number of predefined sentences, and then comparing each entry in the candidate set of potential sentences to said speech query to determine a matching recognized sentence.

46. The system of claim 45, wherein said speech utterance is compared against said candidate set of potential sentences by examining noun phrases.

47. The system of claim 45, wherein said candidate set of potential sentences are determined in part by a context dictionary loaded by said sentence recognition circuit in response to an operating environment presented by said system to a user.

48. The system of claim 44, wherein said speech utterance is processed by a natural language engine.

49. The system of claim 44, wherein environment variables experienced by the user within the web browser program are used for recognizing said speech query, such that said environmental variables vary in accordance with a web page being viewed by the user or a selection within a web page made by the user.

50. The website of claim 44, wherein said list of items correspond to topics associated with an interactive lesson tutorial.

51. The system of claim 44, wherein said limited speech data content does not include complete speech observation vectors which must be derived from said limited speech data content and input to said speech recognition routine before said speech utterance can be recognized.

52. The system of claim 44 wherein said limited speech data content comprises speech data that is transmitted continuously while the user is speaking and until silence is detected.

53. A method of interacting with a web-connected server using a client browser program, the method comprising the steps of:

(a) receiving speech data associated with a speech utterance articulated by a user of the client platform, said speech data being characterized by a limited speech data content to reduce processing and transmission latencies; and

(b) completing recognition of said speech utterance using said limited speech data content to generate a recognized speech query at the web-connected server in real-time; and

(c) presenting one or more web pages to the user client web browser program, such that data content for said one or more web pages transmitted to the client browser program is controlled by said recognized speech query;

(d) allocating signal processing functions required to generate said recognized speech query between a client platform and the server computing system as needed based on computing resources available to said client and server computing systems respectively.

54. The method of claim **53** further including a step: performing a natural language processing operation to compare a limited set of phrases extracted from said recognized speech query with a separate set of phrases extracted from predefined valid queries from users.

55. The method of claim **53** further including a step: providing an interactive electronic character who provides suggestions for queries which the user can articulate.

56. The system of claim **55**, further including a step: configuring said interactive character agent to engage in a dialog of successive questions and answers with the user during an interactive session.

57. The method of claim **53**, further including a step: presenting an interactive character agent to the user in real-time in response to a spoken help command presented while interacting with any web page maintained by the server computing system.

58. The method of claim **53**, further including a step: configuring said web page as a single page to a browser to allow a user to ask questions concerning any item identified in said database within said single page.

59. The method of claim **53**, further including a step: forming a concatenation of words and/or phrases derived from said speech query and using said concatenation as a search query for a database.

60. A method of presenting information from a set of one or more web pages associated with a server interacting through a browser program with a client platform, the method comprising the steps of:

(a) partially processing a speech utterance at the client platform to generate limited data content speech data, said limited data content speech data being configured to reduce processing and transmission latencies; and

(b) completing processing of said speech utterance using said limited speech data content to generate a recognized speech query at the server; and

(c) presenting content for the set of one or more web pages to the browser program, under control of said recognized speech query;

(d) allocating signal processing functions required to generate said recognized speech query between a client platform and the server computing system as needed based on computing resources available to said client and server computing systems respectively.

61. The method of claim **60**, wherein said limited speech data content does not include complete speech observation vectors which must be derived from said limited speech data content and input to said speech recognition routine before said speech utterance can be recognized.

62. The method of claim **60**, wherein the server computing system transfers speech related data for the web page using a hypertext transfer protocol (HTTP).

63. The method of claim **60** further including a step: dynamically changing a speech recognition grammar based on input provided by a user to selections available within said web page.

64. A speech-enabled internet server computing system comprising:

a receiving routine executing on the server computing system for receiving speech data associated with a user speech-based query, said speech data being characterized by a data content that is substantially inadequate by itself for permitting recognition of words articulated in said speech query; and

a speech recognition routine executing on the server computing system for completing recognition of said speech query using said speech data and said data content to generate a recognized speech query;

wherein signal processing functions required to generate said recognized speech query can be allocated between a client platform and the server computing system as needed based on computing resources available to said client platform and server computing system respectively;

a natural language routine executing on the server computing system and configured to process said recognized speech query to generate a natural language result in real-time;

a web page having a list of items, at least some of said list of items being selectable by a user based on said natural language result;

a database coupled to the server computing system for storing predefined answers which correspond to content for said list of items on said web page.

65. The speech-enabled internet server computing system of claim **64**, wherein said web page contains links to other web pages which can be selected by speech queries.

66. The speech-enabled internet server computing system of claim **64**, wherein said web page is a single page configured to allow a user to ask questions concerning any item identified in said database within said single page.

67. The speech-enabled internet server computing system of claim **64** wherein any and all of said list of items are selectable in a single screen.

68. The speech enabled internet server computing system of claim **67**, wherein any and all of said list of items are selectable without scrolling through said web page.

69. A speech-enabled internet server computing system comprising:

a receiving routine executing on the server computing system for receiving speech data associated with a user speech-based query, said speech data being characterized by a data content that is substantially inadequate by itself for permitting recognition of words articulated in said speech query; and

a speech recognition routine executing on the server computing system for completing recognition of said speech query using said speech data and said data content to generate a recognized speech query;

wherein signal processing functions required to generate said recognized speech query can be allocated between a client platform and the server computing system as

needed based on computing resources available to said client platform and server computing system respectively;

a natural language routine executing on the server computing system and configured to process said recognized speech query to generate a natural language result based on an analysis of a selected limited set of phrases presented in said recognized speech query;

wherein said selected limited set of phrases are configured so that said natural language engine can generate said natural language result in real-time;

a web page having a list of items, at least some of said list of items being selectable by a user based on said natural language result;

a database coupled to the server computing system for storing content pertaining to said list of items on said web page.

70. The speech-enabled internet server computing system of claim **69** wherein said selected limited set of phrases include combinations of words and phrases contained in a grammar used by said speech recognition routine.

71. The speech-enabled internet server computing system of claim **69** wherein said selected limited set of phrases are generated dynamically from the recognized speech query.

72. The speech-enabled internet server computing system of claim **69** wherein said natural language engine compares said selected limited set of phrases to a set of phrases contained in predefined answers.

73. The speech-enabled internet server computing system of claim **69** wherein said natural language engine result is a single best answer.

74. A speech-enabled internet server computing system comprising:

a receiving routine executing on the server computing system for receiving speech data associated with a user speech-based query, said speech data being characterized by a data content that is substantially inadequate by itself for permitting recognition of words articulated in said speech query; and

a speech recognition routine executing on the server computing system for completing recognition of said speech query using said speech data and said data content to generate a recognized speech query;

wherein signal processing functions required to generate said recognized speech query can be allocated between a client platform and the server computing system as needed based on computing resources available to said client platform and server computing system respectively;

a natural language routine executing on the server computing system and configured to process said recognized speech query to generate a natural language result based on an analysis of a selected limited set of phrases presented in said recognized speech query;

wherein said selected limited set of phrases are configured so that said natural language engine can generate said natural language result and a response can be provided to said user speech-based query in real-time;

a web page having a list of items, at least some of said list of items being selectable by a user based on said natural language result;

a database coupled to the server computing system for storing content pertaining to said list of items on said web page;

an electronic conversational agent adapted to interact with the user and mimic behavior of a human agent through a native language interactive real-time dialog session with the user.

75. The speech-enabled internet server computing system of claim **74**, wherein said electronic conversational agent is presented within a client browser.

76. The speech-enabled internet server computing system of claim **74**, wherein said electronic conversational agent is a visual character on a screen.

77. The speech-enabled internet server computing system of claim **74**, wherein said electronic conversational agent is configured to articulate suggestions to the user for appropriate speech queries.

78. The speech-enabled internet server computing system of claim **74**, wherein said electronic conversational agent is adapted to have configurable perception parameters which are adjusted and tailored to said content pertaining to said list of items.

* * * * *

EXHIBIT 4

US007277854B2

(12) **United States Patent**
Bennett et al.

(10) **Patent No.: US 7,277,854 B2**
(45) **Date of Patent: Oct. 2, 2007**

(54) **SPEECH RECOGNITION SYSTEM INTERACTIVE AGENT**

(75) Inventors: **Ian M. Bennett**, Palo Alto, CA (US); **Bandi Ramesh Babu**, Anantapur (IN); **Kishor Morkhandikar**, Gulbarga (IN); **Pallaki Gururaj**, Bangalore (IN)

(73) Assignee: **Phoenix Solutions, Inc**, Palo Alto, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 82 days.

(21) Appl. No.: **11/031,633**

(22) Filed: **Jan. 7, 2005**

(65) **Prior Publication Data**

US 2005/0144004 A1 Jun. 30, 2005

Related U.S. Application Data

(63) Continuation of application No. 10/684,357, filed on Oct. 10, 2003, which is a continuation of application No. 09/439,145, filed on Nov. 12, 1999, now Pat. No. 6,633,846.

(51) **Int. Cl.**
G10L 15/18 (2006.01)
G06F 17/20 (2006.01)
G06F 17/28 (2006.01)

(52) **U.S. Cl.** **704/257**; 704/8; 707/5

(58) **Field of Classification Search** 704/270, 704/270.1, 275, 8, 9, 257; 707/3, 4, 5
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,473,904 A 9/1984 Suehiro et al.
4,587,670 A 5/1986 Levinson et al.
4,783,803 A 11/1988 Baker et al.
4,785,408 A 11/1988 Britton et al.
4,852,170 A 7/1989 Bordeaux
4,868,750 A * 9/1989 Kucera et al. 704/8
4,914,590 A 4/1990 Loatman et al.
4,991,094 A 2/1991 Fagan et al.
4,991,217 A 2/1991 Garrett et al.
5,068,789 A 11/1991 van Vliembergen
5,146,405 A 9/1992 Church
5,157,727 A 10/1992 Schloss

(Continued)

FOREIGN PATENT DOCUMENTS

EP 1094388 4/2001

(Continued)

OTHER PUBLICATIONS

B. Arons, "The Design of Audio Servers and Toolkits for Supporting Speech in the User Interface," believed to be published in: Journal of the American Voice I/O Society, pp. 27-41, Mar. 1991.

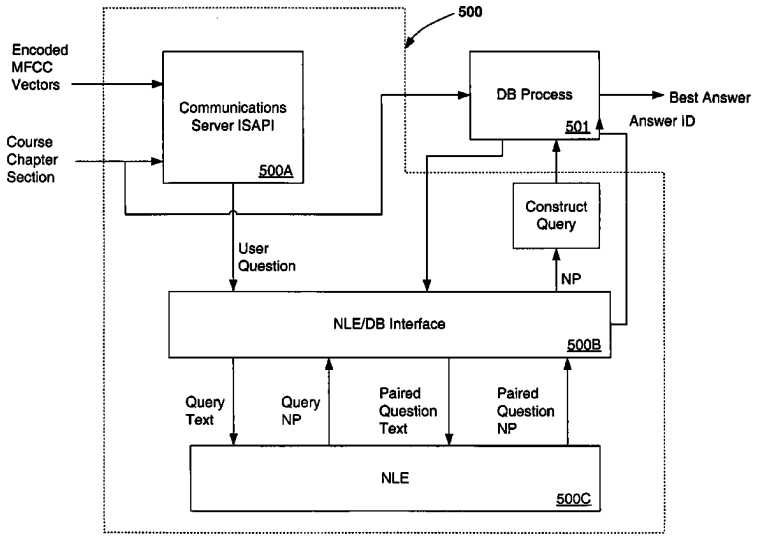
(Continued)

Primary Examiner—Martin Lerner
(74) *Attorney, Agent, or Firm*—J. Nicholas Gross

(57) **ABSTRACT**

A speech recognition system includes distributed processing across a client and server for recognizing a spoken query by a user. A number of different speech models for different languages are used to support and detect a language spoken by a user. In some implementations an interactive electronic agent responds in the user's language to facilitate a real-time, human like dialogue.

29 Claims, 31 Drawing Sheets



U.S. PATENT DOCUMENTS

5,231,670	A		7/1993	Goldhor et al.
5,265,014	A		11/1993	Haddock et al.
5,278,980	A		1/1994	Pedersen et al.
5,293,584	A		3/1994	Brown et al.
5,371,901	A		12/1994	Reed et al.
5,384,892	A		1/1995	Strong
5,454,106	A		9/1995	Burns et al.
5,475,792	A		12/1995	Stanford et al.
5,500,920	A	*	3/1996	Kupiec 704/270.1
5,509,104	A		4/1996	Lee et al.
5,513,298	A		4/1996	Stanford et al.
5,524,169	A		6/1996	Cohen et al.
5,540,589	A		7/1996	Waters
5,553,119	A		9/1996	McAllister et al.
5,602,963	A		2/1997	Bissonnette et al.
5,615,296	A		3/1997	Stanford et al.
5,625,814	A		4/1997	Luciw
5,652,897	A		7/1997	Linebarger et al.
5,668,854	A		9/1997	Minakami et al.
5,675,707	A		10/1997	Gorin et al.
5,680,511	A		10/1997	Baker et al.
5,680,628	A		10/1997	Carus et al.
5,694,592	A		12/1997	Driscoll
5,727,950	A		3/1998	Cook et al.
5,730,603	A	*	3/1998	Harless 434/308
5,737,485	A		4/1998	Flanagan et al.
5,758,023	A		5/1998	Bordeaux
5,758,322	A		5/1998	Rongley
5,765,130	A		6/1998	Nguyen
5,774,841	A		6/1998	Salazar et al.
5,794,193	A		8/1998	Gorin
5,802,251	A		9/1998	Cohen et al.
5,802,526	A		9/1998	Fawcett et al.
5,819,220	A		10/1998	Sarukkai et al.
5,836,771	A		11/1998	Ho et al.
5,860,063	A		1/1999	Gorin et al.
5,865,626	A		2/1999	Beattie et al.
5,867,817	A		2/1999	Catallo et al.
5,873,062	A		2/1999	Hansen et al.
5,878,406	A	*	3/1999	Noyes 706/55
5,884,302	A		3/1999	Ho
5,905,773	A		5/1999	Wong
5,915,236	A		6/1999	Gould et al.
5,933,822	A	*	8/1999	Braden-Harder et al. 707/5
5,934,910	A		8/1999	Ho et al.
5,956,675	A		9/1999	Setlur et al.
5,956,683	A		9/1999	Jacobs et al.
5,960,394	A		9/1999	Gould et al.
5,960,399	A		9/1999	Barclay et al.
5,963,940	A	*	10/1999	Liddy et al. 707/5
5,978,756	A		11/1999	Walker et al.
5,987,410	A		11/1999	Kellner et al.
5,995,918	A		11/1999	Kendall et al.
5,995,928	A		11/1999	Nguyen et al.
6,006,221	A	*	12/1999	Liddy et al. 707/5
6,009,387	A		12/1999	Ramaswamy et al.
6,018,736	A		1/2000	Gilai et al.
6,021,384	A		2/2000	Gorin et al.
6,023,697	A		2/2000	Bates et al.
6,029,124	A		2/2000	Gillick et al.
6,032,111	A		2/2000	Mohri
6,035,275	A		3/2000	Brode et al.
6,044,266	A		3/2000	Kato
6,044,337	A		3/2000	Gorin et al.
6,061,646	A		5/2000	Martino et al.
6,078,914	A		6/2000	Redfern
6,081,774	A		6/2000	de Hita et al.
6,088,692	A		7/2000	Driscoll
6,105,023	A		8/2000	Callan
6,112,176	A		8/2000	Goldenthal et al.
6,119,087	A		9/2000	Kuhn et al.
6,122,613	A		9/2000	Baker
6,125,284	A		9/2000	Moore et al.
6,125,341	A		9/2000	Raud et al.
6,138,089	A		10/2000	Guberman
6,141,640	A		10/2000	Moo
6,144,848	A		11/2000	Walsh et al.
6,144,938	A		11/2000	Surace et al.
6,173,261	B1		1/2001	Arai et al.
6,173,279	B1		1/2001	Levin et al.
6,175,634	B1		1/2001	Graumann
6,178,404	B1		1/2001	Hambleton et al.
6,182,038	B1		1/2001	Balakrishnan et al.
6,182,068	B1		1/2001	Culliss
6,185,535	B1		2/2001	Hedin et al.
6,192,110	B1		2/2001	Abella et al.
6,195,636	B1		2/2001	Crupi et al.
6,216,013	B1		4/2001	Moore et al.
6,226,610	B1		5/2001	Keiller et al.
6,233,559	B1		5/2001	Balakrishnan
6,243,679	B1		6/2001	Mohri et al.
6,246,986	B1		6/2001	Ammicht et al.
6,246,989	B1		6/2001	Polcyn
6,256,607	B1		7/2001	Digalakis et al.
6,269,153	B1		7/2001	Carpenter et al.
6,269,336	B1		7/2001	Ladd et al.
6,278,973	B1		8/2001	Chung et al.
6,292,767	B1		9/2001	Jackson et al.
6,292,781	B1		9/2001	Urs et al.
6,311,159	B1		10/2001	Van Tichelen et al.
6,314,402	B1		11/2001	Monaco et al.
6,327,343	B1		12/2001	Epstein et al.
6,327,561	B1		12/2001	Smith et al.
6,327,568	B1		12/2001	Joost
6,336,090	B1		1/2002	Chou et al.
6,363,349	B1		3/2002	Urs et al.
6,374,219	B1		4/2002	Jiang
6,374,226	B1		4/2002	Hunt et al.
6,381,594	B1		4/2002	Eichstaedt et al.
6,389,389	B1		5/2002	Meunier et al.
6,393,403	B1		5/2002	Majaniemi
6,408,272	B1		6/2002	White et al.
6,411,926	B1		6/2002	Chang
6,418,199	B1		7/2002	Perrone
6,427,063	B1		7/2002	Cook et al.
6,434,524	B1		8/2002	Weber
6,434,529	B1		8/2002	Walker et al.
6,446,064	B1		9/2002	Livowsky
6,453,020	B1		9/2002	Hughes et al.
6,453,290	B1		9/2002	Jochumson
6,499,011	B1		12/2002	Souvignier et al.
6,499,013	B1		12/2002	Weber
6,510,411	B1		1/2003	Norton et al.
6,513,037	B1		1/2003	Ruber et al.
6,519,562	B1		2/2003	Phillips et al.
6,522,725	B2		2/2003	Kato
6,532,444	B1		3/2003	Weber
6,567,778	B1		5/2003	Chao Chang et al.
6,574,597	B1		6/2003	Mohri et al.
6,584,464	B1		6/2003	Warthen
6,594,269	B1		7/2003	Polcyn
6,594,348	B1		7/2003	Bjurstrom et al.
6,601,026	B2		7/2003	Appelt et al.
6,614,885	B2		9/2003	Polcyn
6,615,172	B1		9/2003	Bennett et al.
6,633,846	B1		10/2003	Bennett et al.
6,647,363	B2	*	11/2003	Claassen 704/1
6,651,043	B2		11/2003	Ammicht
6,665,640	B1		12/2003	Bennett et al.
6,665,644	B1		12/2003	Kanevsky et al.
6,681,206	B1		1/2004	Gorin et al.
6,697,780	B1		2/2004	Beutnagel et al.
6,738,743	B2		5/2004	Sharma et al.
6,742,021	B1		5/2004	Halverson et al.

6,785,647	B2	8/2004	Hutchinson	
6,785,654	B2	8/2004	Cyr et al.	
6,823,308	B2	11/2004	Keiller et al.	
6,842,767	B1	1/2005	Partovi et al.	
6,856,960	B1	2/2005	Dragosh et al.	
6,862,713	B1	3/2005	Kraft et al.	
6,871,179	B1	3/2005	Kist et al.	
6,901,399	B1 *	5/2005	Corston et al.	707/6
6,922,733	B1	7/2005	Kuiken et al.	
6,940,953	B1	9/2005	Eberle et al.	
6,941,273	B1	9/2005	Loghmani et al.	
6,944,586	B1	9/2005	Harless et al.	
6,961,954	B1	11/2005	Maybury et al.	
6,964,012	B1	11/2005	Zirngibl et al.	
6,965,864	B1	11/2005	Thrift et al.	
6,965,890	B1	11/2005	Dey et al.	
7,003,463	B1	2/2006	Maes et al.	
7,020,609	B2	3/2006	Thrift et al.	
7,031,925	B1	4/2006	Goldberg	
7,058,573	B1	6/2006	Murveit et al.	
7,082,397	B2	7/2006	Cohen et al.	
2001/0013001	A1	8/2001	Brown et al.	
2001/0016813	A1	8/2001	Brown et al.	
2001/0032083	A1	10/2001	Van Cleven	
2001/0056346	A1	12/2001	Ueyama et al.	
2002/0032566	A1	3/2002	Tzirkel-Hancock et al.	
2002/0046023	A1	4/2002	Fujii et al.	
2002/0059068	A1	5/2002	Rose et al.	
2002/0059069	A1	5/2002	Hsu et al.	
2002/0086269	A1	7/2002	Shpiro	
2002/0087325	A1	7/2002	Lee et al.	
2002/0087655	A1	7/2002	Bridgman et al.	
2002/0091527	A1	7/2002	Shiau	
2003/0191625	A1	10/2003	Gorin et al.	
2005/0091056	A1	4/2005	Surace et al.	
2005/0131704	A1	6/2005	Dragosh et al.	

FOREIGN PATENT DOCUMENTS

EP	1096471	5/2001
WO	WO98/11534	3/1998
WO	WO99/48011	9/1999
WO	WO99/50830	10/1999
WO	WO 00/14727	3/2000
WO	WO 00/17854	3/2000
WO	WO 00/20962	4/2000
WO	WO 00/21075	4/2000
WO	WO 00/21232	4/2000
WO	WO 00/22610	4/2000
WO	WO 00/30072	5/2000
WO	WO 00/30287	5/2000
WO	WO 00/68823	11/2000
WO	WO 01/16936	3/2001
WO	WO 01/18693	3/2001
WO	WO 01/26093	4/2001
WO	WO 01/78065	10/2001
WO	WO 01/95312	12/2001
WO	WO 02/03380	1/2002

OTHER PUBLICATIONS

L.R. Bahl et al., "A Maximum Likelihood Approach to Continuous Speech Recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-5, pp. 179-190, Mar. 1983. (12 pages).

J.H. Baker, "The Dragon System—An Overview," IEEE Trans. on ASSP Processing, ASSP-23(1): Feb. 24-29, 1975.

L.E. Baum et al., "Statistical Inference for Probabilistic Functions for Finite State Markov Chains," The Annals of Mathematical Statistics, 37: 1554-1563, 1966.

L.E. Baum et al., "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains," The Annals of Mathematical Statistics, 1970, vol. 41, No. 1, pp. 164-171.

L.E. Baum, "An Inequality and Associated Maximation Technique in Statistical Estimation for Probabilistic Functions of Markov Processes," Inequalities-III, pp. 1-8, 1972.

I. Bennett, "A Study of Speech Compression Using Analog Time Domain Sampling techniques," A Dissertation Submitted to the Dept. Of Electrical Engineering and the Committee on Graduate Studies of Stanford University, May 1975, pp. 16-32; 76-111.

V. Digilakis et al., "Product-Code Vector Quantization of Cepstral Parameters for Speech Recognition over the WWW," believed to be published in: Proc. ICSLP '98, 4 pages, 1998.

V. Digilakis et al., "Quantization of Cepstral Parameters for Speech Recognition over the World Wide Web," believed to be published in: IEEE Journal on Selected Areas of Communications, 22 pages, 1999.

J.L. Flanagan, "Speech Analysis Synthesis and Perception," 2nd edition, Springer-Verlag Berlin, 1972, pp. 1-53.

G.D. Forney, "The Viterbi Algorithm," Proceedings of the IEEE, vol. 73, pp. 268-278, Mar. 1973.

A. Gersho et al., "Vector Quantization and Compression," Kluwer Academic Publishers, 1991, pp. 309-340.

D. Giuliani et al., "Training of HMM with Filtered Speech Material for Hands-Free Recognition," believed to be published in: Proceedings of ICASSP '99, Phoenix, USA, 4 pages, 1999.

T. Hazen et al., "Recent Improvements in an Approach to Segment-Based Automatic Language Identification," believed to be published in: Proceedings of the 1994 International Conference on Spoken Language Processing, Yokohama, Japan, pp. 1883-1886, Sep. 1994.

D. House, "Spoken-Language Access to Multimedia (SLAM): A Multimodal Interface to the World-Wide Web," Masters Thesis, Oregon Graduate Institute, Department of Computer Science & Engineering, 59 pages, Apr. 1995.

R.A. Hudson, "Word Grammar," Blackman Inc., Cambridge, MA, 1984, pp. 1-14; 41-42; 76-90; 94-98; 106-109; 211-220.

F. Jelinek et al, "Continuous Speech Recognition: Statistical Methods," Handbook of Statistics, vol. 2, P.R. Krishnaiah, Ed. Amsterdam, The Netherlands, North-Holland, 1982, pp. 549-573.

L. Julia et al., "http://www.speech.sri.com/demos/ atis.html," believed to be published in: Proceedings AAAI'97: Stanford, pp. 72-76, Jul. 1997.

R. Lau et al, "Webgalaxy-Integrating Spoken Language and Hypertext Navigation," believed to be published in: in Kokkinakis, G. et al., (Eds.) Eurospeech '97, Proceedings of the 5th European Conference on Speech Communication and Technology, Rhodes (Greece), Sep. 22-25, 1997. pp. 883-886, 1997.

P. Lieberman, "Intonation, Perception and Language," Research Monograph No. 38, MIT Press, Cambridge, Mass., 1967, pp. 5-37.

B. Lin et al., "A Distributed Architecture for Cooperative Spoken Dialogue Agents with Coherent Dialogue State and History," believed to be published in: IEEE Automatic Speech Recognition and Understanding Workshop, Keystone, Colorado, USA, 4 pages, Dec. 1999.

B. Lu et al., "Scalability Issues in the Real Time Protocol (RTP)," Project Report for CPSC 663 (Real Time Systems), Dept. of Computer Science, Texas A & M University, 19 pages, 1999.

J. Makhoul et al., "Vector Quantization in Speech Coding," Proceedings of the IEEE, vol. 73, No. 11, Nov. 1985, pp. 1551-1588. (38 pages).

H. Melin, "On Word Boundary Detection in Digit-Based Speaker Verification," believed to be published in: Workshop on Speaker Recognition and its Commercial and Forensic Applications (RLA2C), Avignon, France, Apr. 20-23, pp. 46-49, 1998.

N. Morgan et al., "Hybrid Neural Network/Hidden Markov Model Systems for Continuous Speech Recognition," Journal of Pattern Recognition and Artificial Intelligence, vol. 7, No. 4, 1993, pp. 899-916.

R. Quirk et al., "A Comprehensive Grammar of English Language," Longman, London and New York, 1985, pp. 245-331.

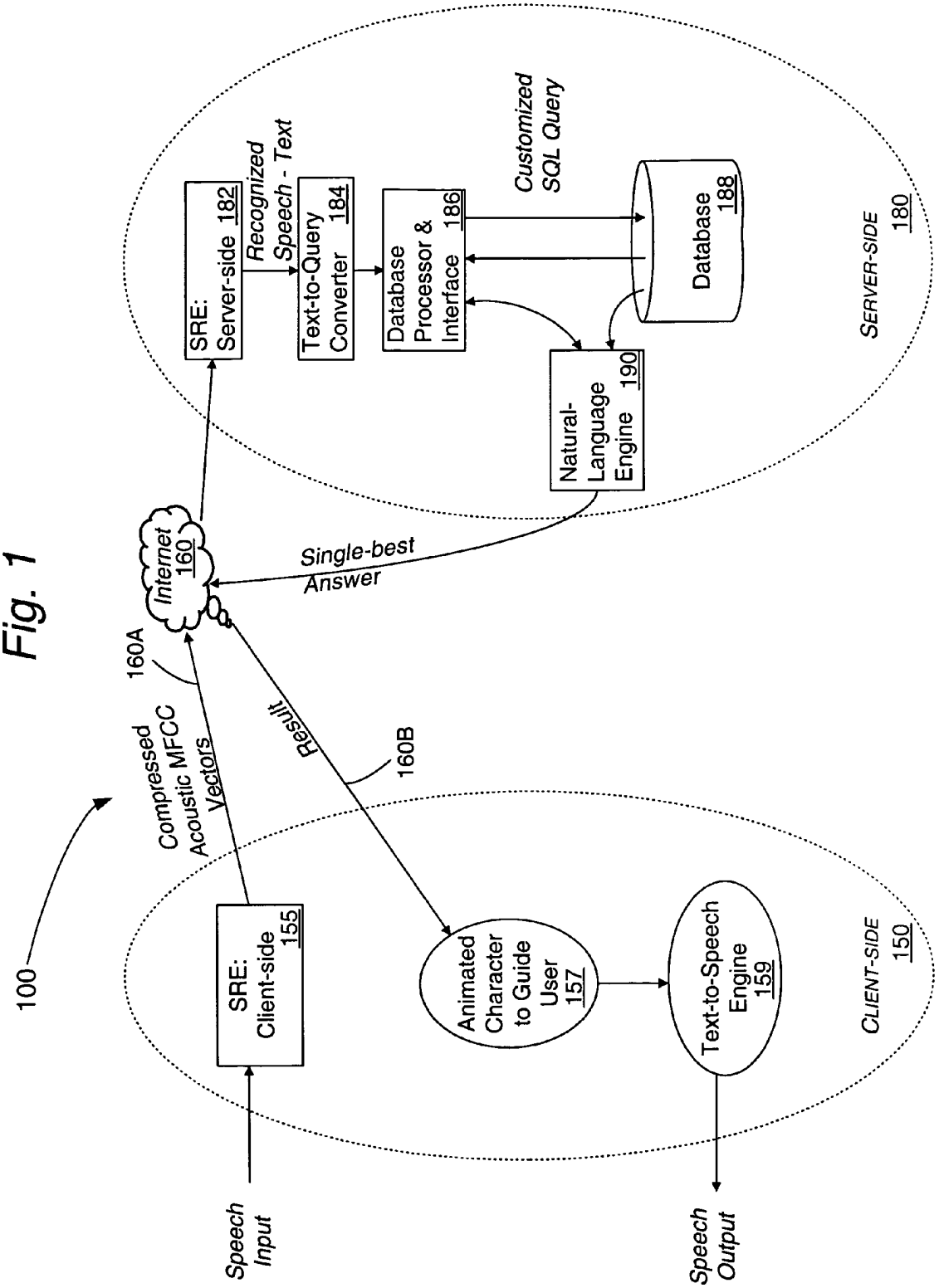
L. Rabiner, "Digital Processing of Speech Signals," Prentice Hall, 1978, pp. 116-171; 355-395.

L. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," Proceedings of the IEEE, vol. 77, No. 2, Feb. 1989, pp. 257-286. (30 pages).

L. Rabiner et al., "Fundamentals of Speech Recognition," Prentice Hall, 1993, pp. 11-68.
G. Ramaswamy et al., "Compression of Acoustic Features for Speech Recognition in Network Environments," believed to be published in: IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 977-980, Jun. 1998.
L. Travis, "Handbook of Speech Pathology," Appleton-Century-Crofts, Inc., 1957, pp. 91-124.
S. Tsakalidis et al., "Efficient Speech Recognition Using Subvector Quantization and Discrete-Mixture HMMs," believed to be published in: Proc. ICASSP '99, 4 pages, 1999.
Agarwal, R., Towards a PURE Spoken Dialogue System for Information Access, believed to be published in *Proceedings of the ACL/EACL Workshop on Interactive Spoken Dialog Systems: Bringing Speech and NLP Together in Real Applications*, Madrid, Spain, 1997, 9 pages.
Ammicht, Egbert et al., "Knowledge Collection for Natural Language Spoken Dialog Systems," believed to be published in *Proc. Eurospeech*, vol. 3, p. 1375-1378, Budapest, Hungary, Sep. 1999, 4 pages.
AT&T Corp., "Network Watson 1.0 System Overview," 1998, 4 pages.
AT&T Corp., "AT&T Watson Advanced Speech Applications Platform," 1996, 3 pages.
AT&T Corp., "AT&T Watson Advanced Speech Applications Platform Version 2.0," 1996, 3 pages.
AT&T Corp., "AT&T Watson Advanced Speech Application Platform Version 2.0," 1996, 8 pages.
Gorin, Allen, "Processing of Semantic Information in Fluently Spoken Language," believed to be published in *Proc. ICSLP*, Philadelphia, PA, Oct. 1996, 4 pages.
Gorin, Allen et al., "How May I Help You," believed to be published in *Proc. IVTTA*, Basking Ridge, NJ, Oct. 1996, 32 pages.
Mohri, Mehryar, "String Matching With Automata," Nordic Journal of Computing, 1997, 15 pages.
Prudential News, "Prudential Pilots Revolutionary New Speech-Based Telephone Customer Service System Developed by AT&T Labs—Company Business and Marketing," Dec. 6, 1999, 3 pages.
Riccardi, Giuseppe et al., "A spoken language system for automated call routing," believed to be published in *Proc. ICASSP '97*, 1997, 4 pages.
Sharp, Douglas, et al., "The Watson Speech Recognition Engine," accepted by ICASSP, 1997, 9 pages.
European Patent Office search report for EP Application No. 00977144, dated Mar. 30, 2005, 5 pages.
Burstein, A. et al. "Using Speech Recognition In A Personal Communications System," *Proceedings of the International Conference on Communications*, Chicago, Illinois, Jun. 14-18, 1992, pp. 1717-1721.
Digalakis, V. et al., "Quantization Of Cepstral Parameters For Speech Recognition Over The World Wide Web," *IEEE Journal on Selected Areas of Communications*, 1999, pp. 82-90.
Kuhn, T. et al., "Hybrid In-Car Speech Recognition For Mobile Multimedia Applications," *Vehicular Technology Conference*, Houston, Texas, May 1999, pp. 2009-2013.
Cox, Richard V. et al., "Speech and Language Processing for Next-Millennium Communications Services," Proceedings of the IEEE, vol. 88, No. 8, Aug. 2000, pp. 1314-1337.
Kuhn, Roland, and Renato De Mori, "The Application of Semantic Classification Trees to Natural Language Understanding," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 17, No. 5, May 1995, pp. 449-460.
Unisys Corp., "Natural Language Speech Assistant (NLSA) Capabilities Overview," NLR 3.0, Aug. 1998, Malvern, PA, 27 pages.
Buo, Finn Dag et al., "Learning to parse spontaneous speech." Believed to be published in ICSLP, 1996, 4 pages.
Golden, J. et al., "Automatic Topic Identification for Two-level Call Routing," Proc. International Conference on Acoustics, Speech and Signal Processing, vol. 1, pp. 509-512, 1999.
Carbonell, J. et al., Dynamic strategy selection in flexible parsing.) In Proceedings of the 19th annual meeting of the ACL (ACL 81), 1981, pp. 143-147.
Seneff, S. et al., "Galaxy-II: a Reference Architecture for Conversational System Development," Believed to be published in Proceedings of ICSLP98 (1998), 4 pages.
Microsoft Internet Developer, "Add Natural Language Search Capabilities to Your Site with English Query," http://www.microsoft.com/mind/0498/equery.asp, 1998, 9 pages.
Carroll, J. et al., "Dialogue Management in Vector-Based Call Routing," Believed to be published In Proceedings of the Annual Meeting of the Association for Computational Linguistics, 1998, pp. 256-262.
Kaiser, Ed., "Robust, Finite-state Parsing for Spoken Language", Student Session of ACL '99, Jun. 1999, pp. 573-578.
Coffman, Daniel et al., Provisional Application for Patent, U.S. Appl. No. 60/117,595, filed Jan. 27, 1999, 111 pages.

* cited by examiner

Fig. 1


100
SRE: Client-side 155
Animated Character to Guide User 157
Text-to-Speech Engine 159
CLIENT-SIDE 150
Speech Input
Speech Output
Compressed MFCC Acoustic Vectors
160A
Result
160B
Internet 160
Single-best Answer
SRE: Server-side 182
Recognized Speech - Text
Text-to-Query Converter 184
Database Processor & Interface 186
Customized SQL Query
Database 188
Natural-Language Engine 190
SERVER-SIDE 180

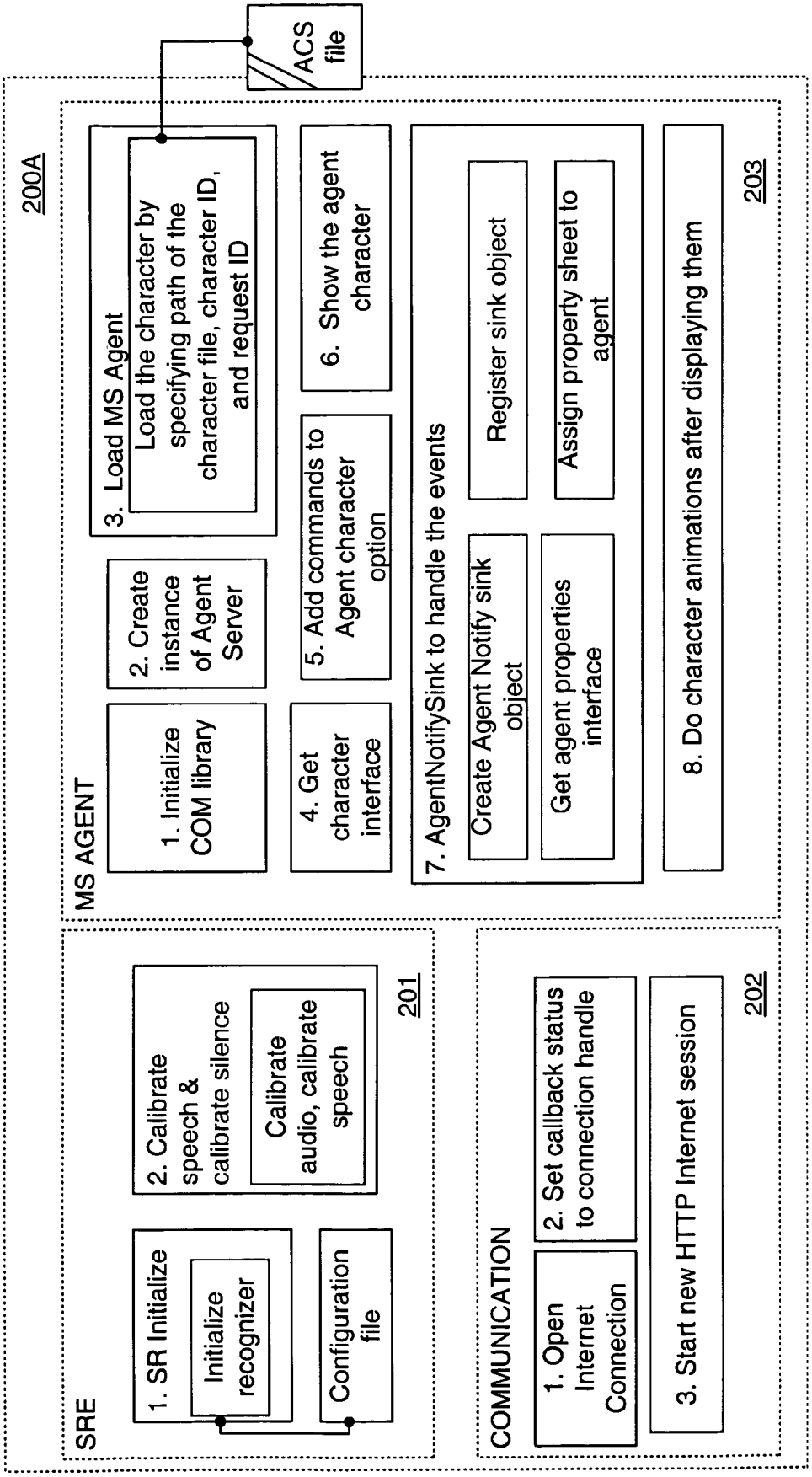


Figure 2A
CLIENT-SIDE SYSTEM LOGIC

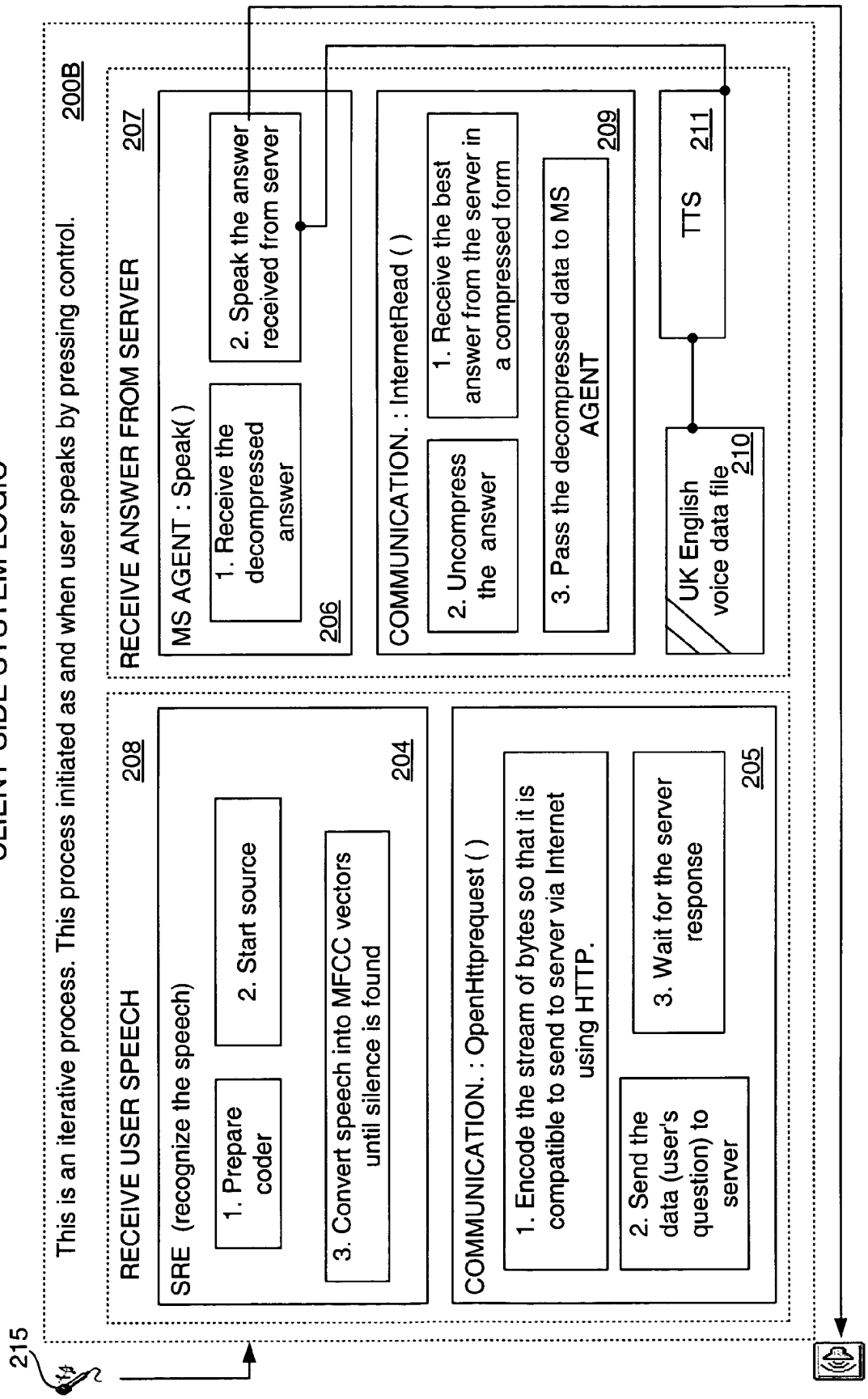


Figure 2B

CLIENT-SIDE SYSTEM LOGIC

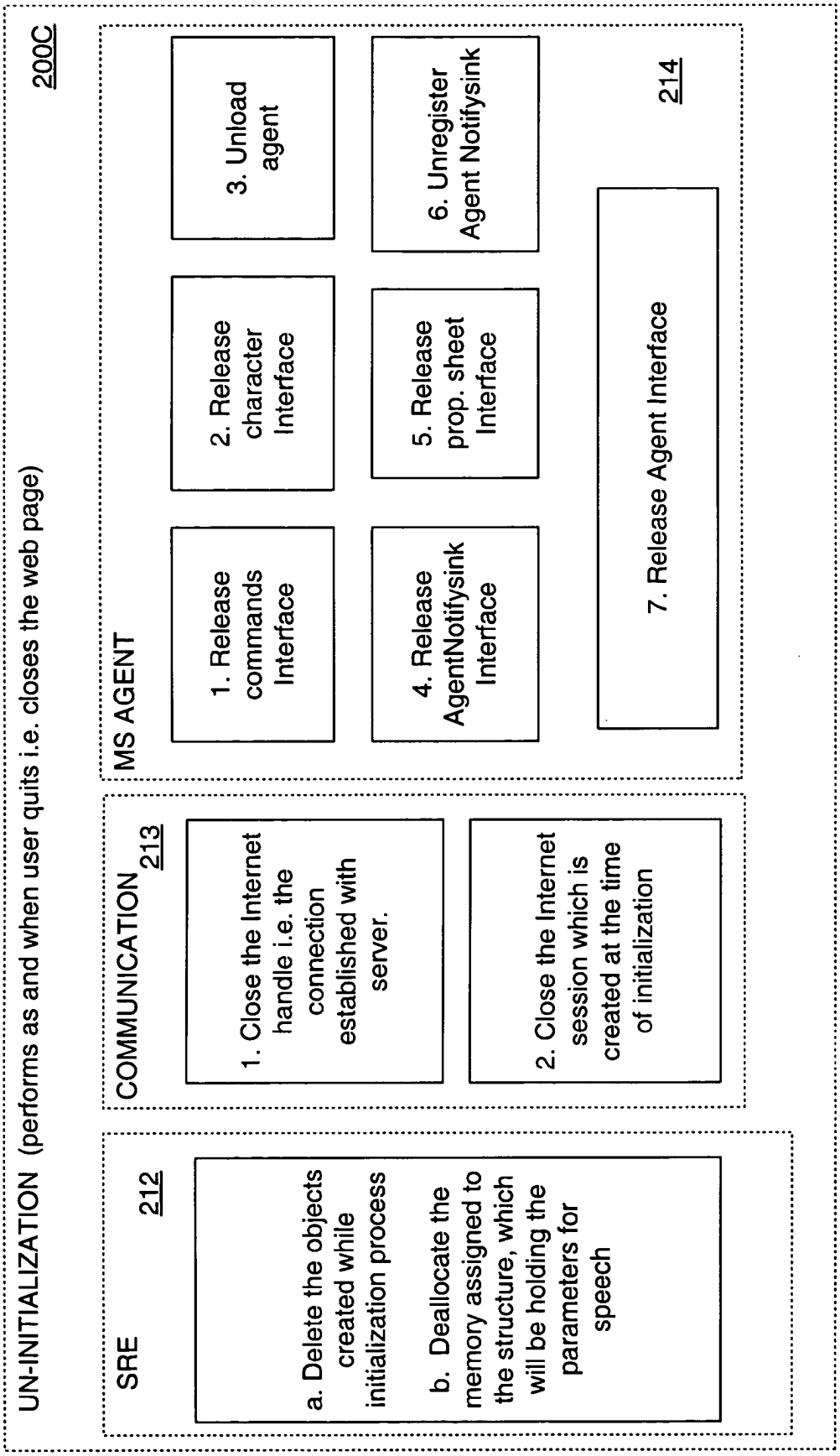


Figure 2C
CLIENT-SIDE SYSTEM LOGIC

Fig. 2D

Client-side Initialization

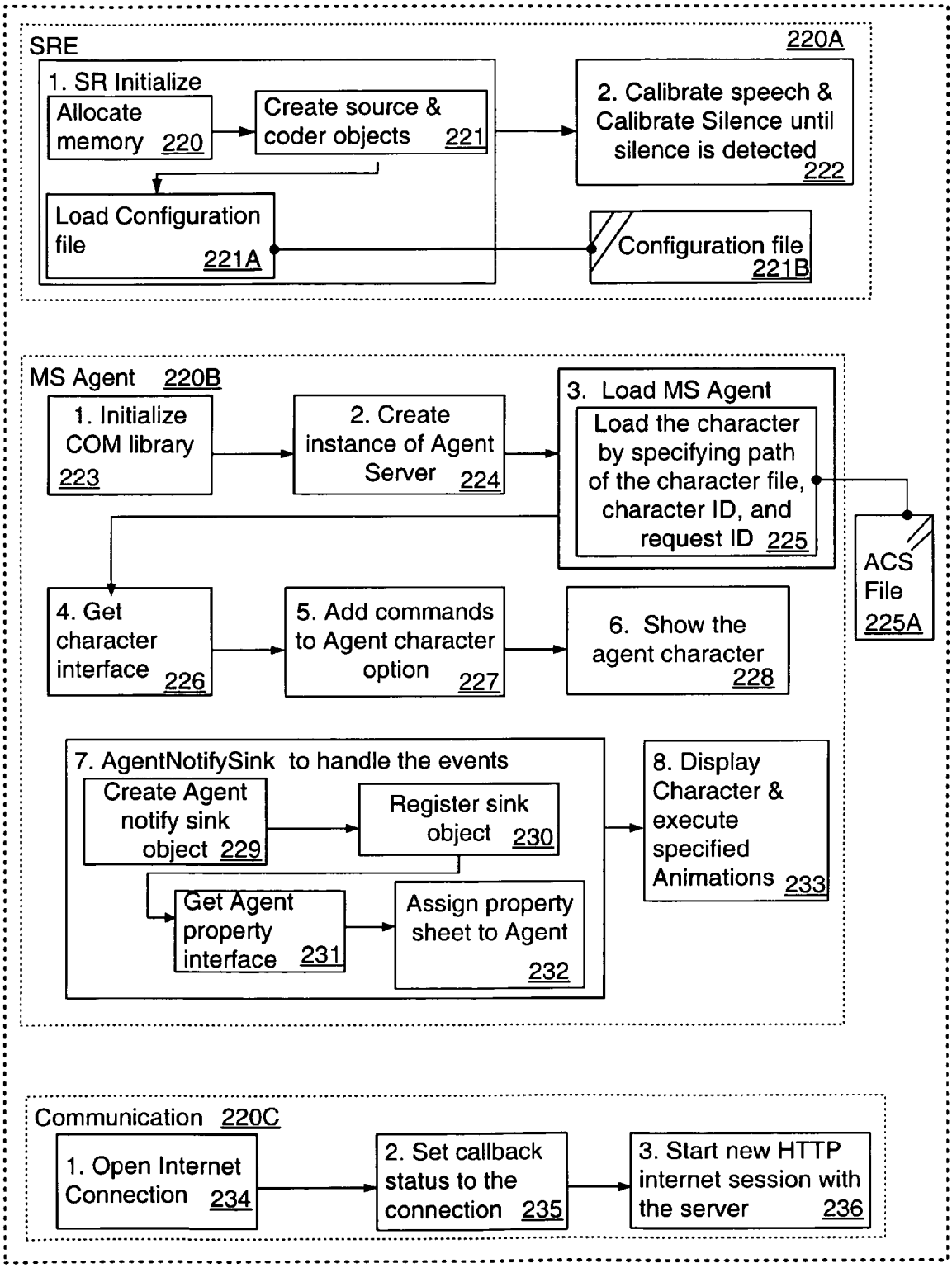


Fig. 3

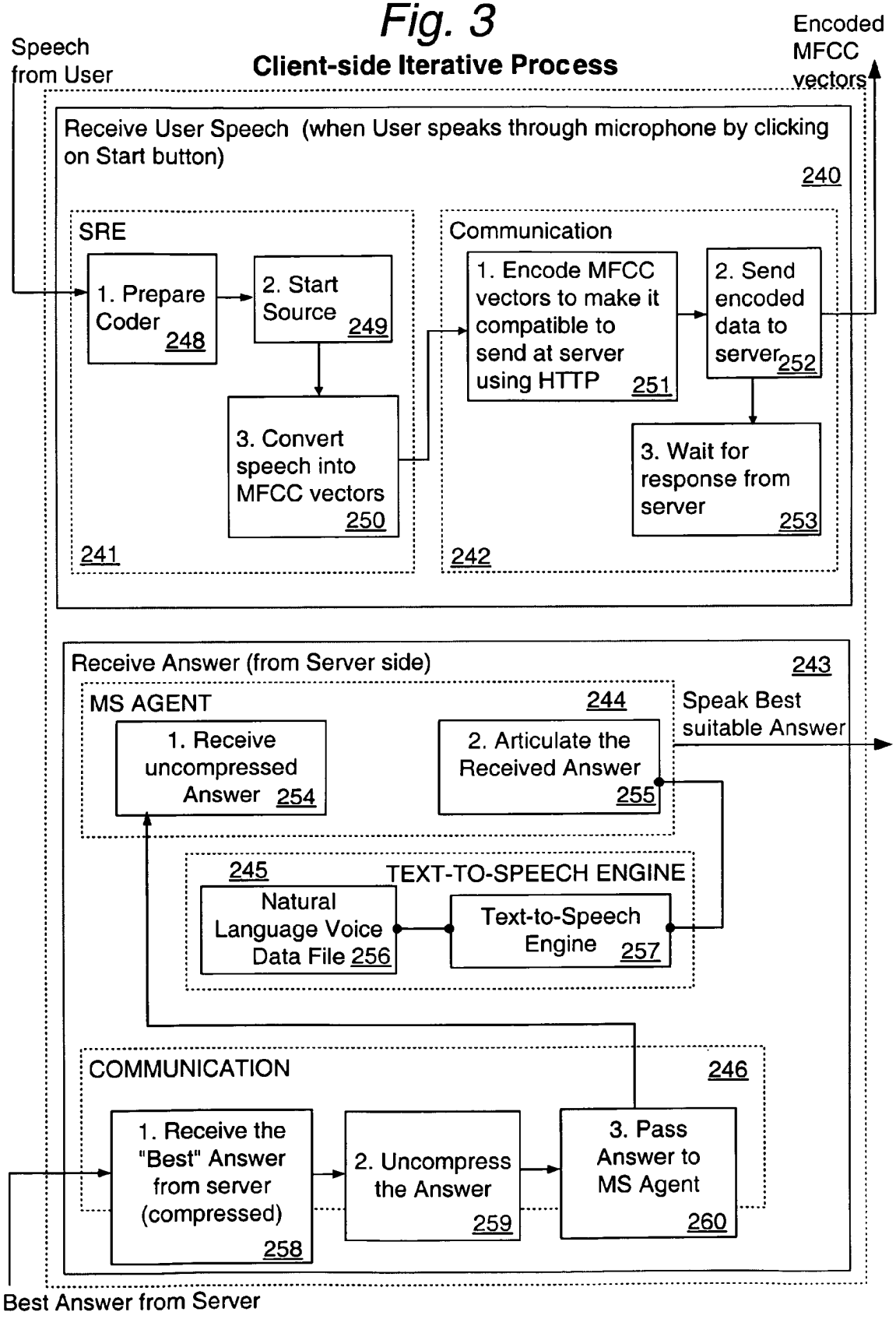

Client-side Iterative Process
Speech from User
Encoded MFCC vectors
Receive User Speech (when User speaks through microphone by clicking on Start button)
240
SRE
1. Prepare Coder 248
2. Start Source 249
3. Convert speech into MFCC vectors 250
241
Communication
1. Encode MFCC vectors to make it compatible to send at server using HTTP 251
2. Send encoded data to server 252
3. Wait for response from server 253
242
Receive Answer (from Server side)
243
MS AGENT
244
1. Receive uncompressed Answer 254
2. Articulate the Received Answer 255
Speak Best suitable Answer
245
TEXT-TO-SPEECH ENGINE
Natural Language Voice Data File 256
Text-to-Speech Engine 257
COMMUNICATION
246
1. Receive the "Best" Answer from server (compressed) 258
2. Uncompress the Answer 259
3. Pass Answer to MS Agent 260
Best Answer from Server

Fig. 4

Client-side Un-Initialization

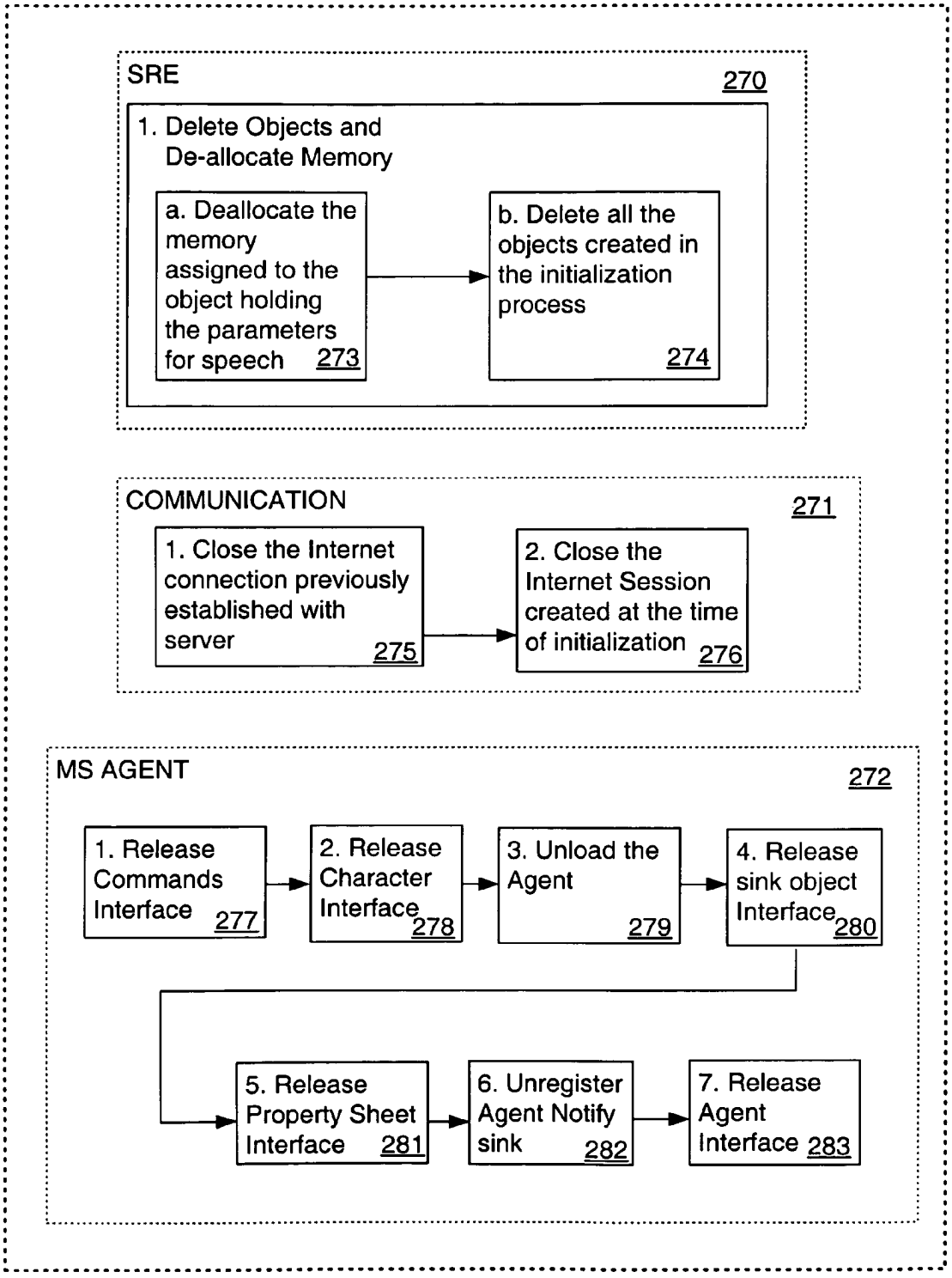
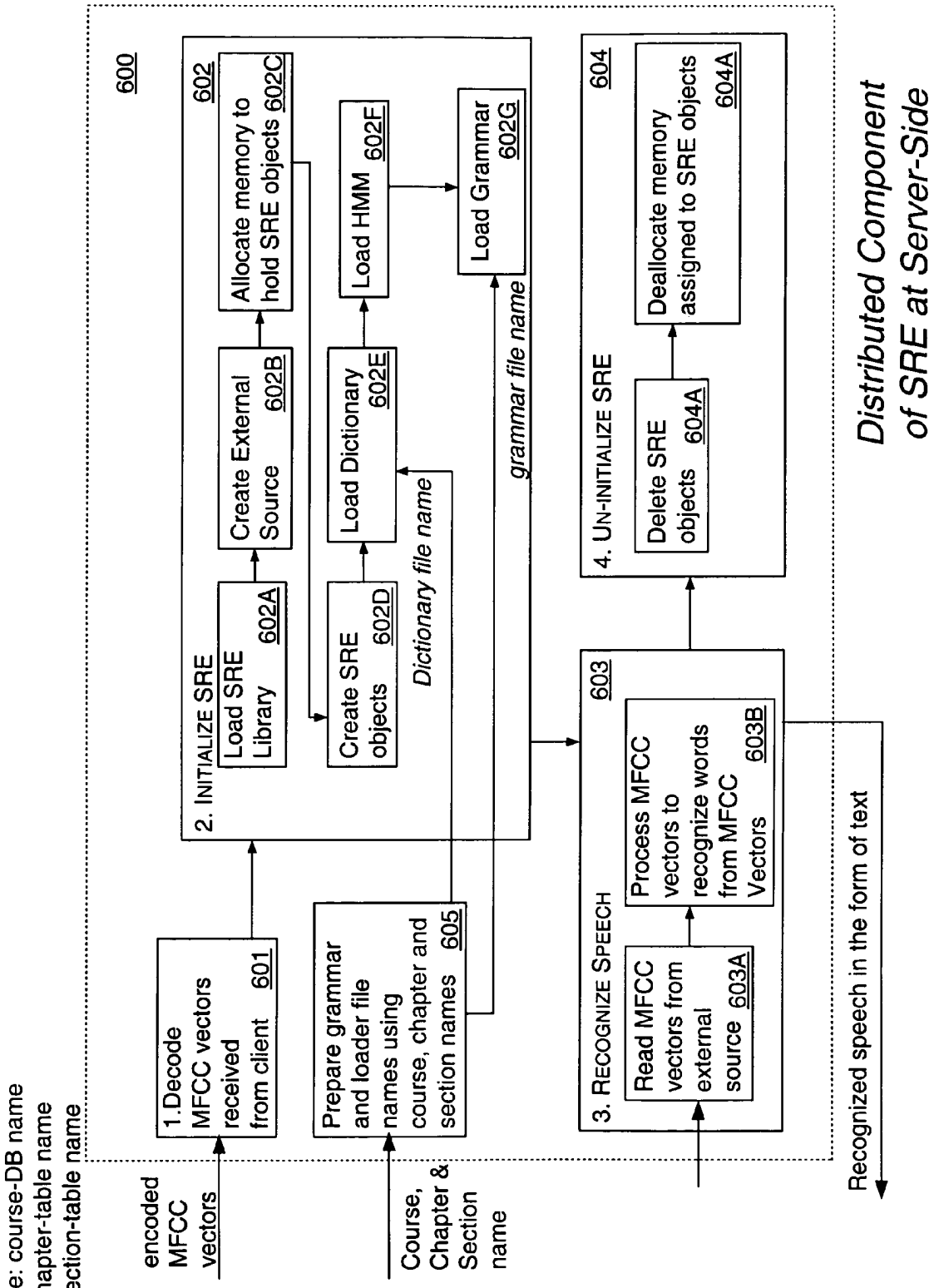


Fig. 4A


600
Distributed Component of SRE at Server-Side
Note: course-DB name
chapter-table name
section-table name
encoded MFCC vectors
Course, Chapter & Section name
Recognized speech in the form of text
1.Decode MFCC vectors received from client 601
Prepare grammar and loader file names using course, chapter and section names 605
2. INITIALIZE SRE 602
Load SRE Library 602A
Create External Source 602B
Allocate memory to hold SRE objects 602C
Create SRE objects 602D
Load Dictionary 602E
Load HMM 602F
Load Grammar 602G
Dictionary file name
grammar file name
3. RECOGNIZE SPEECH 603
Read MFCC vectors from external source 603A
Process MFCC vectors to recognize words from MFCC Vectors 603B
4. UN-INITIALIZE SRE 604
Delete SRE objects 604A
Deallocate memory assigned to SRE objects 604A

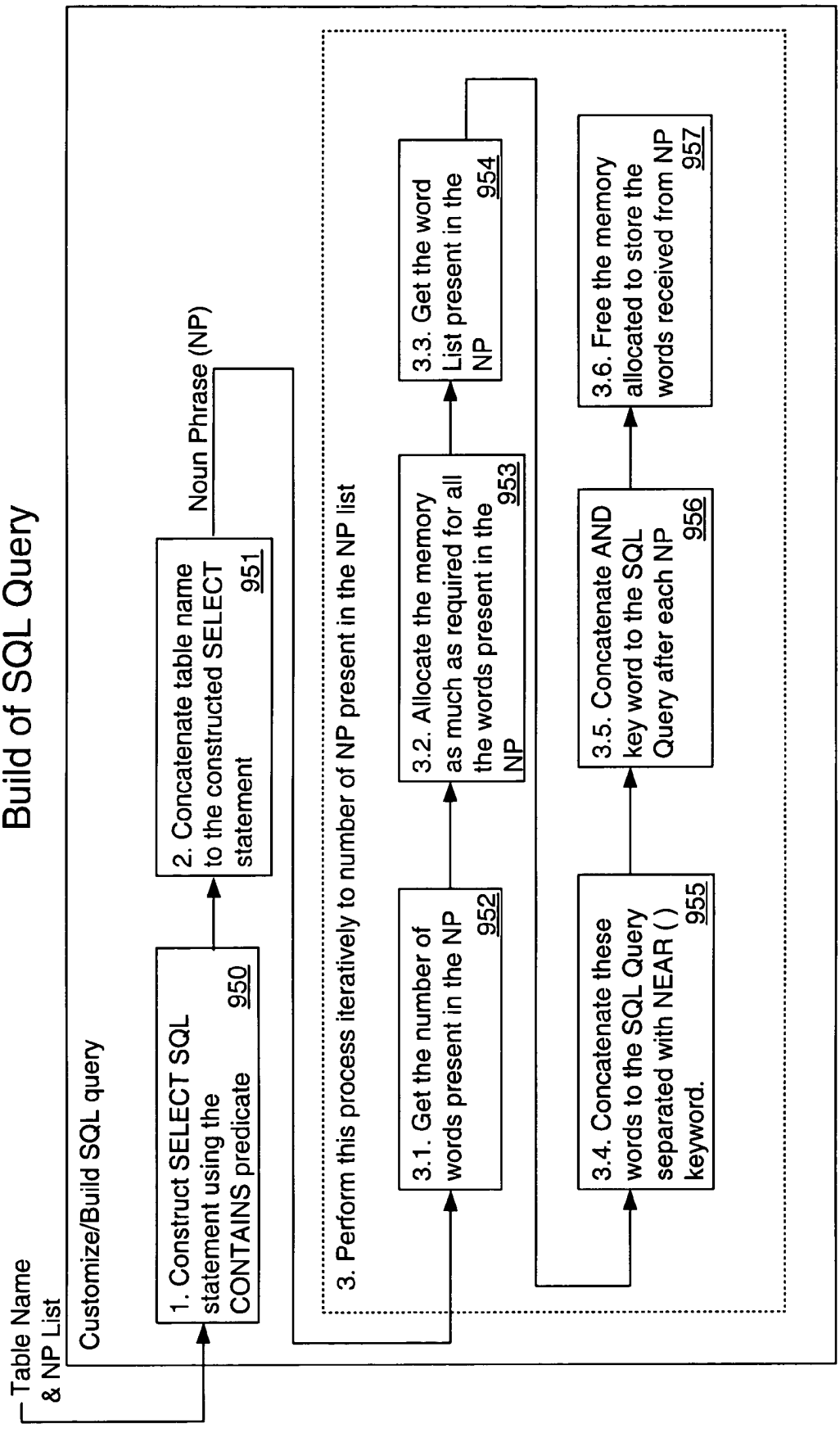


Fig. 4B
Build of SQL Query

Fig. 4C

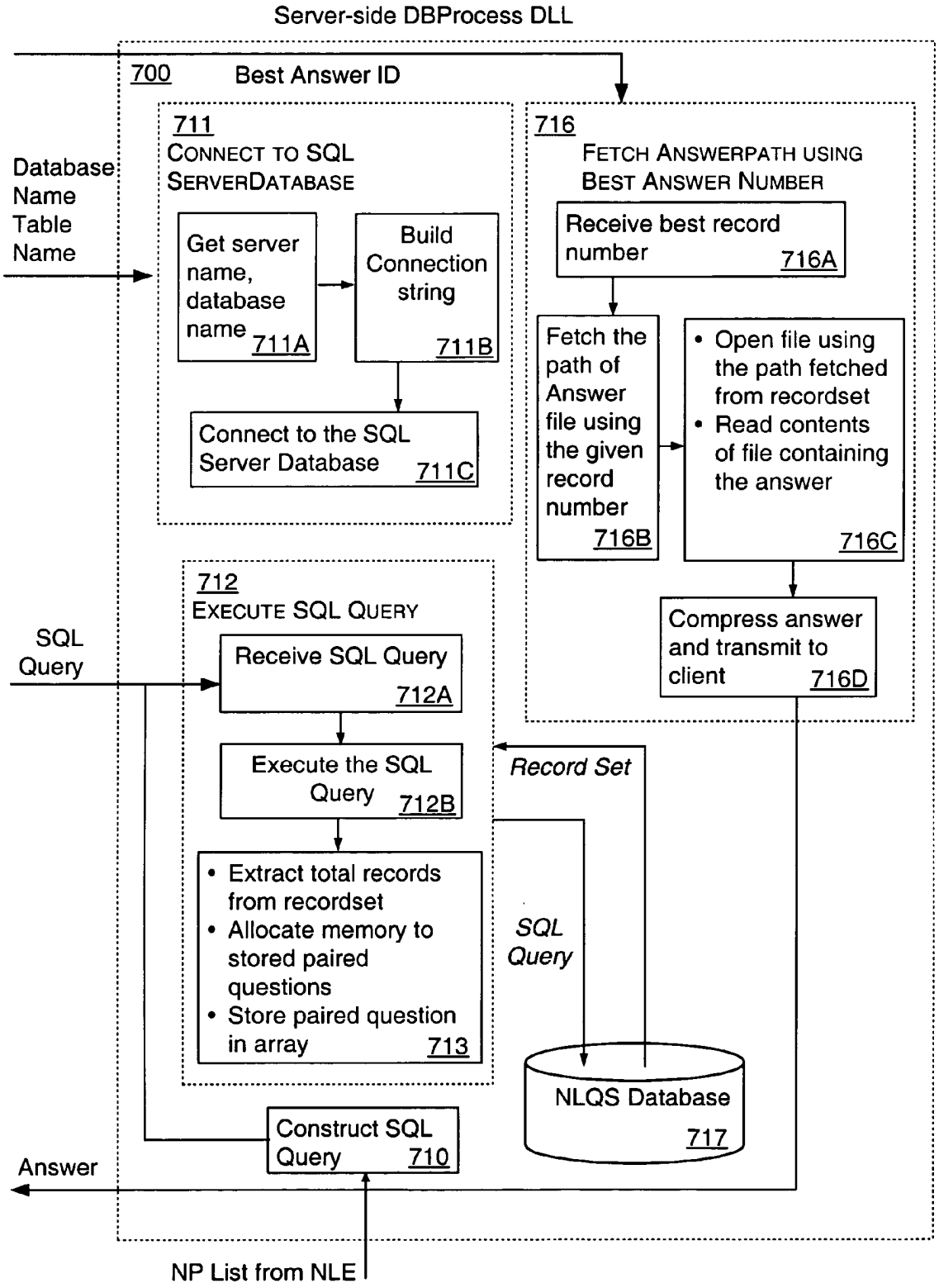

Server-side DBProcess DLL
700
Best Answer ID
711
CONNECT TO SQL SERVERDATABASE
Database Name
Table Name
Get server name, database name 711A
Build Connection string 711B
Connect to the SQL Server Database 711C
716
FETCH ANSWERPATH USING BEST ANSWER NUMBER
Receive best record number 716A
Fetch the path of Answer file using the given record number 716B
• Open file using the path fetched from recordset
• Read contents of file containing the answer 716C
Compress answer and transmit to client 716D
712
EXECUTE SQL QUERY
SQL Query
Receive SQL Query 712A
Execute the SQL Query 712B
Record Set
• Extract total records from recordset
• Allocate memory to stored paired questions
• Store paired question in array 713
SQL Query
NLQS Database 717
Construct SQL Query 710
Answer
NP List from NLE

Fig. 4D

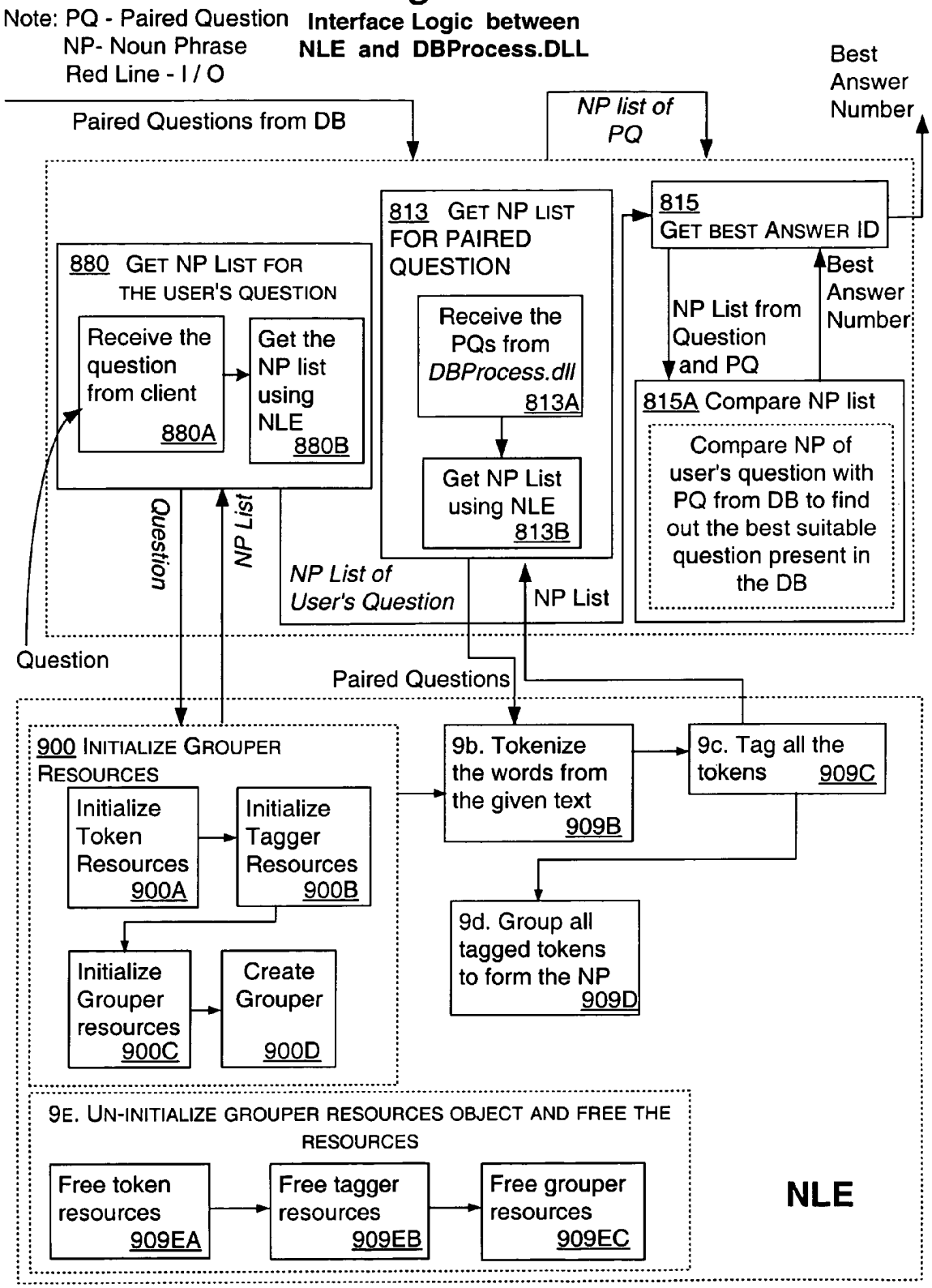

Note: PQ - Paired Question
NP- Noun Phrase
Red Line - I / O
Interface Logic between NLE and DBProcess.DLL
Paired Questions from DB
NP list of PQ
Best Answer Number
880 GET NP LIST FOR THE USER'S QUESTION
Receive the question from client 880A
Get the NP list using NLE 880B
813 GET NP LIST FOR PAIRED QUESTION
Receive the PQs from DBProcess.dll 813A
Get NP List using NLE 813B
815 GET BEST ANSWER ID
NP List from Question and PQ
Best Answer Number
815A Compare NP list
Compare NP of user's question with PQ from DB to find out the best suitable question present in the DB
Question
NP List
NP List of User's Question
NP List
Question
Paired Questions
900 INITIALIZE GROUPER RESOURCES
Initialize Token Resources 900A
Initialize Tagger Resources 900B
Initialize Grouper resources 900C
Create Grouper 900D
9b. Tokenize the words from the given text 909B
9c. Tag all the tokens 909C
9d. Group all tagged tokens to form the NP 909D
9E. UN-INITIALIZE GROUPER RESOURCES OBJECT AND FREE THE RESOURCES
Free token resources 909EA
Free tagger resources 909EB
Free grouper resources 909EC
NLE

Fig. 5

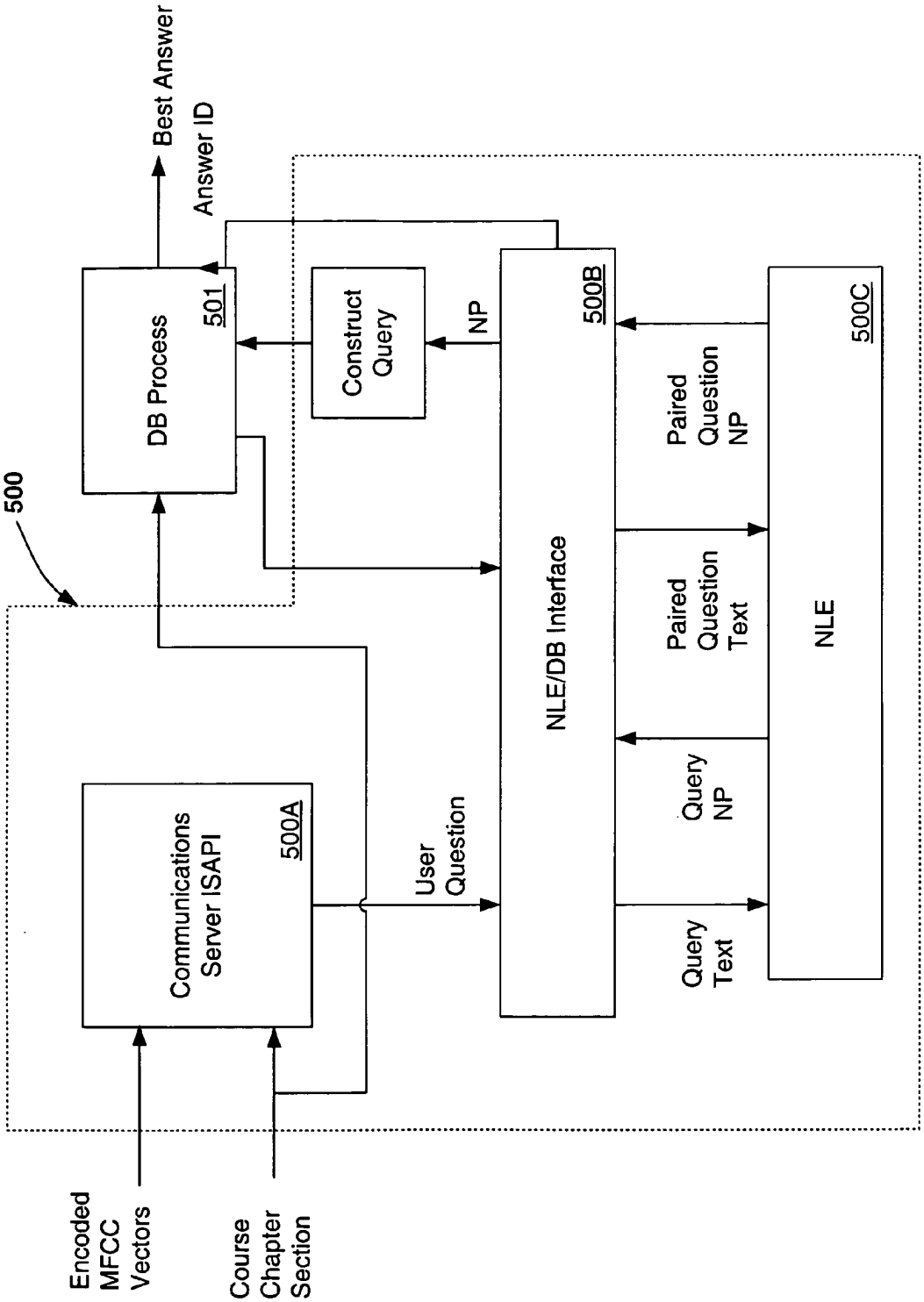

Best Answer
Answer ID
500
DB Process
501
Construct Query
NP
NLE/DB Interface
500B
Paired Question NP
Paired Question Text
Query NP
Query Text
NLE
500C
Communications Server ISAPI
500A
User Question
Encoded MFCC Vectors
Course Chapter Section

Fig.6

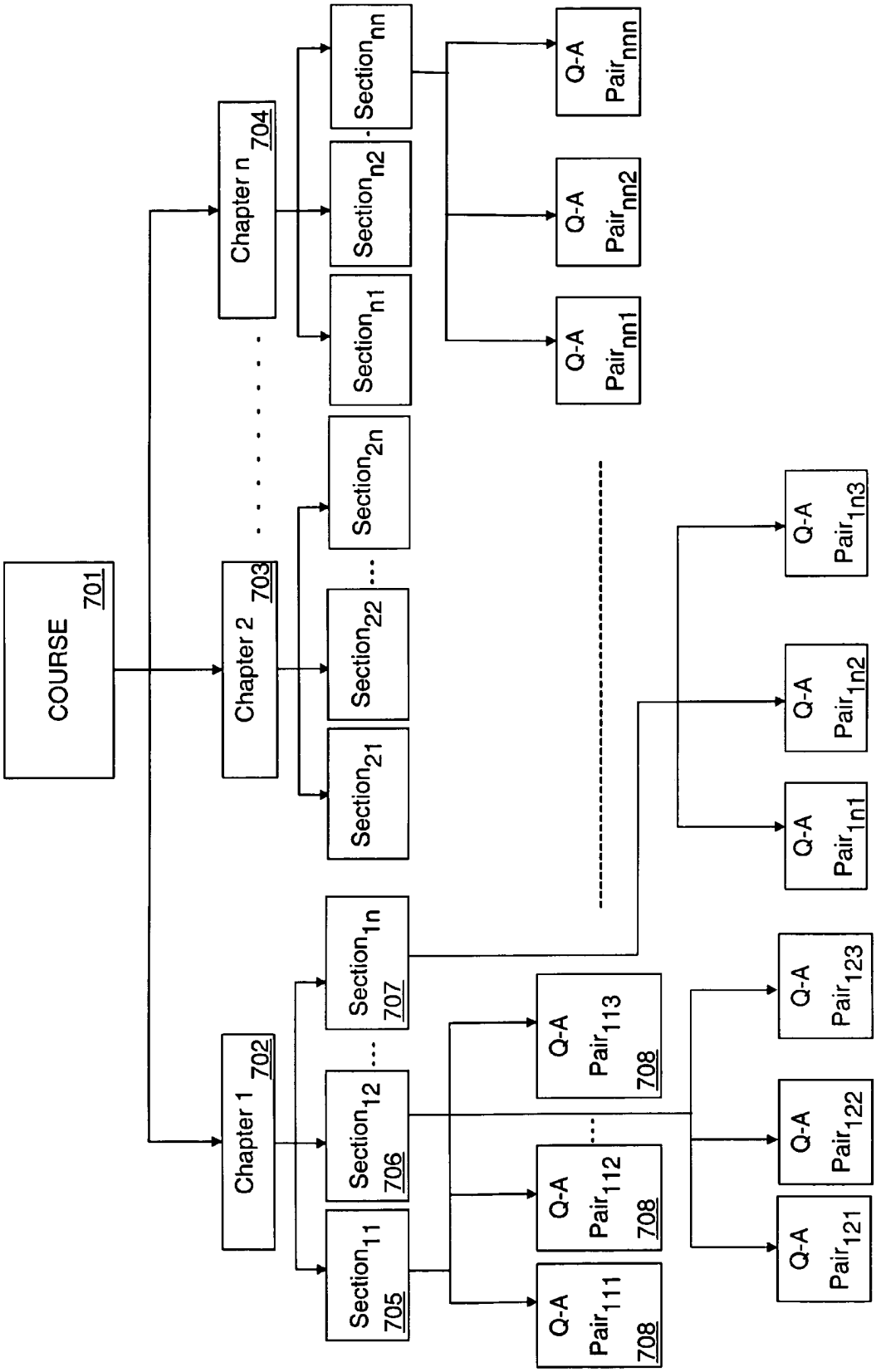

COURSE 701
Chapter 1 702
Chapter 2 703
Chapter n 704
Section11 705
Section12 706
Section1n 707
Section21
Section22
Section2n
Sectionn1
Sectionn2
Sectionnn
Q-A Pair111 708
Q-A Pair112 708
Q-A Pair113 708
Q-A Pair121
Q-A Pair122
Q-A Pair123
Q-A Pair1n1
Q-A Pair1n2
Q-A Pair1n3
Q-A Pairnn1
Q-A Pairnn2
Q-A Pairnnn

Fig. 7A

FIELD NAME 701A	DATA TYPE 702A	SIZE 703A	NULL 704A	PRIMARY KEY 705A	INDEXED? 706A
ChapterName 707A	Varchar	255	No	No	Yes
SectionName 708A	Varchar	255	No	No	Yes

Fig. 7B

FIELD NAME 720	DATA TYPE 721	SIZE 722	NULL 723	PRIMARY KEY 724	INDEXED? 725
Chapter_ID 726	Integer		No	Yes	Yes
Answer_ID 727	Char	5	No	UNIQUE	Yes
Section_Name 728	Varchar	255	No	UNIQUE	Yes
Answer_Title 729	Varchar	255	Yes	No	Yes
PairedQuestion 730	Text	16	No	No	Yes (Full-Text)
AnswerPath 731	Varchar	255	No	No	Yes
Creator 732	Varchar	50	No	No	Yes
Date_of_Creation 733	Date	-	No	No	Yes
Date_of_Modification 734	Date	-	No	No	Yes

Fig. 7C

Field	720	Description	735
AnswerID	727	An integer - automatically incremented for user convenience	
Section_Name	728	Name of section to which the particular record belongs. This field along with AnswerID has to be made primary key	
Answer_Title	729	A short description of the answer	
PairedQuestion	730	Contains one or more combinations of questions for the related answer whose path is stored in the next column AnswerPath	
AnswerPath	731	Contains the path of text file, which contains the answer to the related questions stored in the previous column	
Creator	732	Name of content creator	
Date_of_Creation	733	Date on which content has been added	
Date_of_Modification	734	Date on which content has been changed or modified	

Fig. 7D

FIELD 740	DATA TYPE 741	SIZE 742	NULL 743	PRIMARY KEY 744	INDEXED 745
Answer_ID 746	Char	5	No	Yes	Yes
Answer_Title 747	Varchar	255	Yes	No	No
PairedQuestion 748	Text	16	No	No	Yes (Full-Text)
Answer_Path 749	Varchar	255	No	No	No
Creator 750	Varchar	50	No	No	No
Date_of_Creation 751	Date	-	No	No	No
Date_of_Modification 752	Date	-	No	No	No

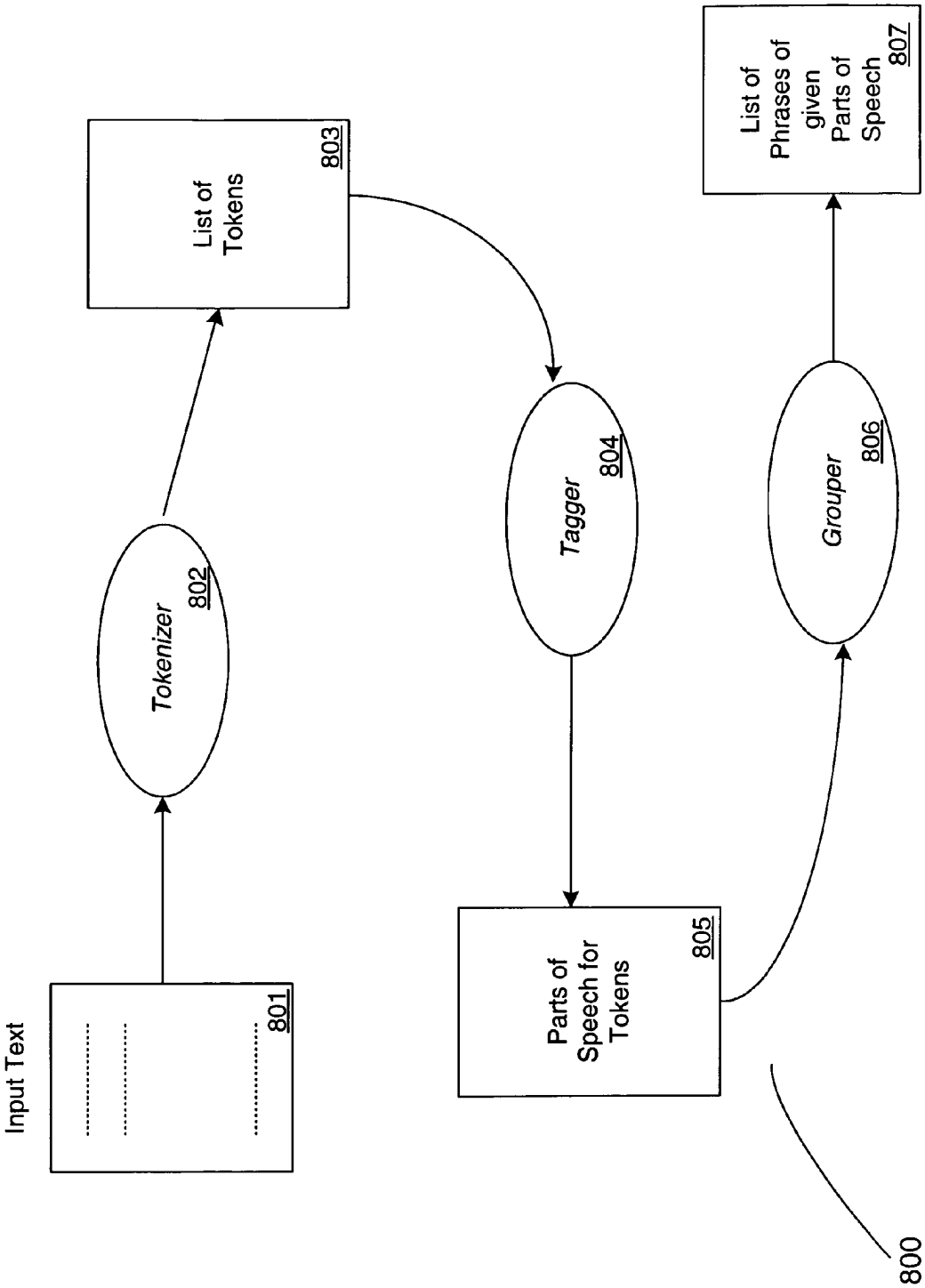

Input Text
801
Tokenizer
802
List of Tokens
803
Tagger
804
Parts of Speech for Tokens
805
Grouper
806
List of Phrases of given Parts of Speech
807
800

Fig. 8

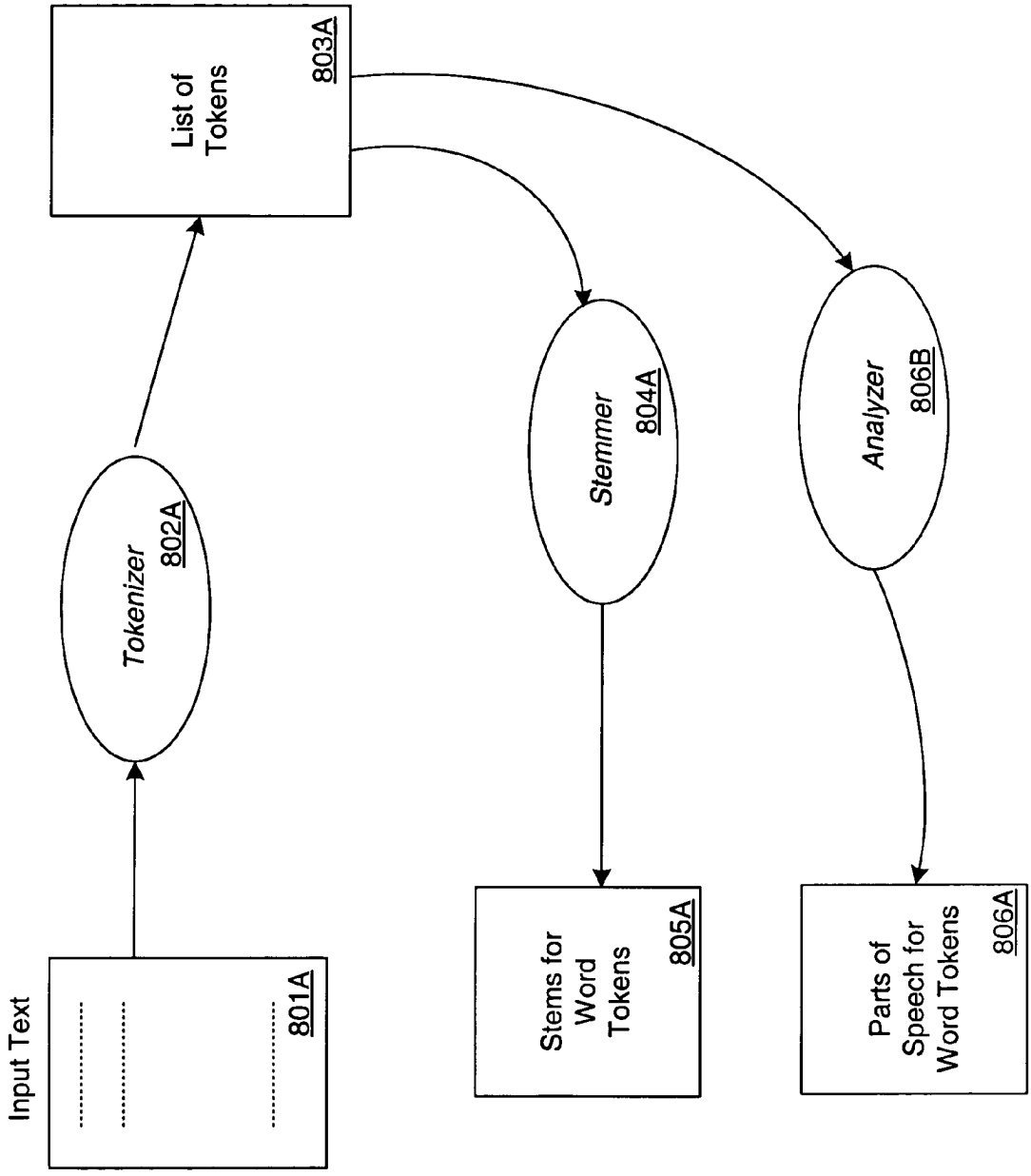

Input Text
801A
Tokenizer
802A
List of Tokens
803A
Stemmer
804A
Analyzer
806B
Stems for Word Tokens
805A
Parts of Speech for Word Tokens
806A

Fig. 9

Fig. 10

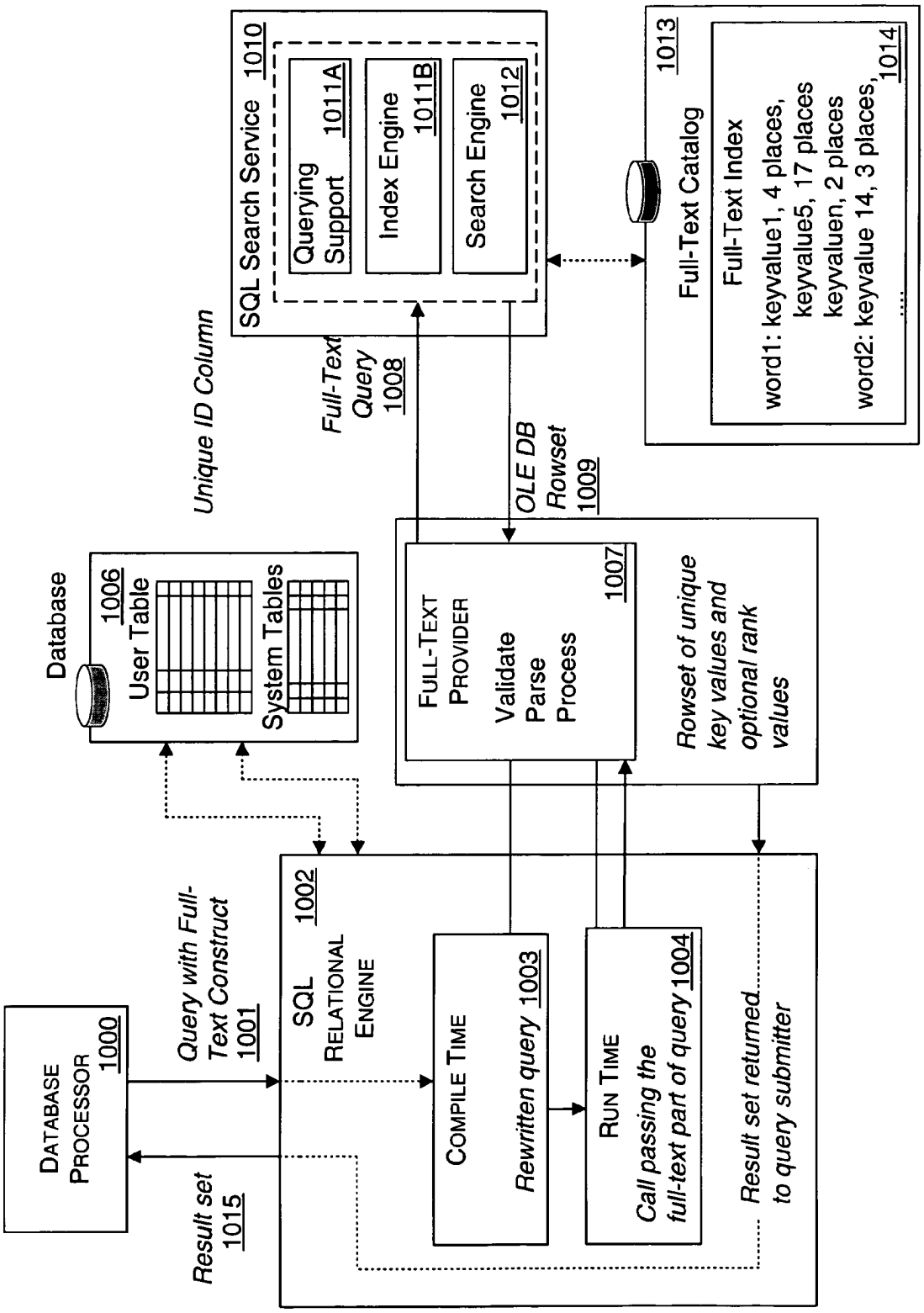

DATABASE PROCESSOR 1000
Query with Full-Text Construct 1001
Result set 1015
SQL RELATIONAL ENGINE 1002
COMPILE TIME
Rewritten query 1003
RUN TIME
Call passing the full-text part of query 1004
Result set returned to query submitter
FULL-TEXT PROVIDER
Validate
Parse
Process
1007
Rowset of unique key values and optional rank values
OLE DB Rowset 1009
Full-Text Query 1008
Unique ID Column
Database
1006
User Table
System Tables
SQL Search Service 1010
Querying Support 1011A
Index Engine 1011B
Search Engine 1012
1013
Full-Text Catalog
Full-Text Index
word1: keyvalue1, 4 places,
keyvalue5, 17 places
keyvaluen, 2 places
word2: keyvalue 14, 3 places,
....
1014

Fig. 11A

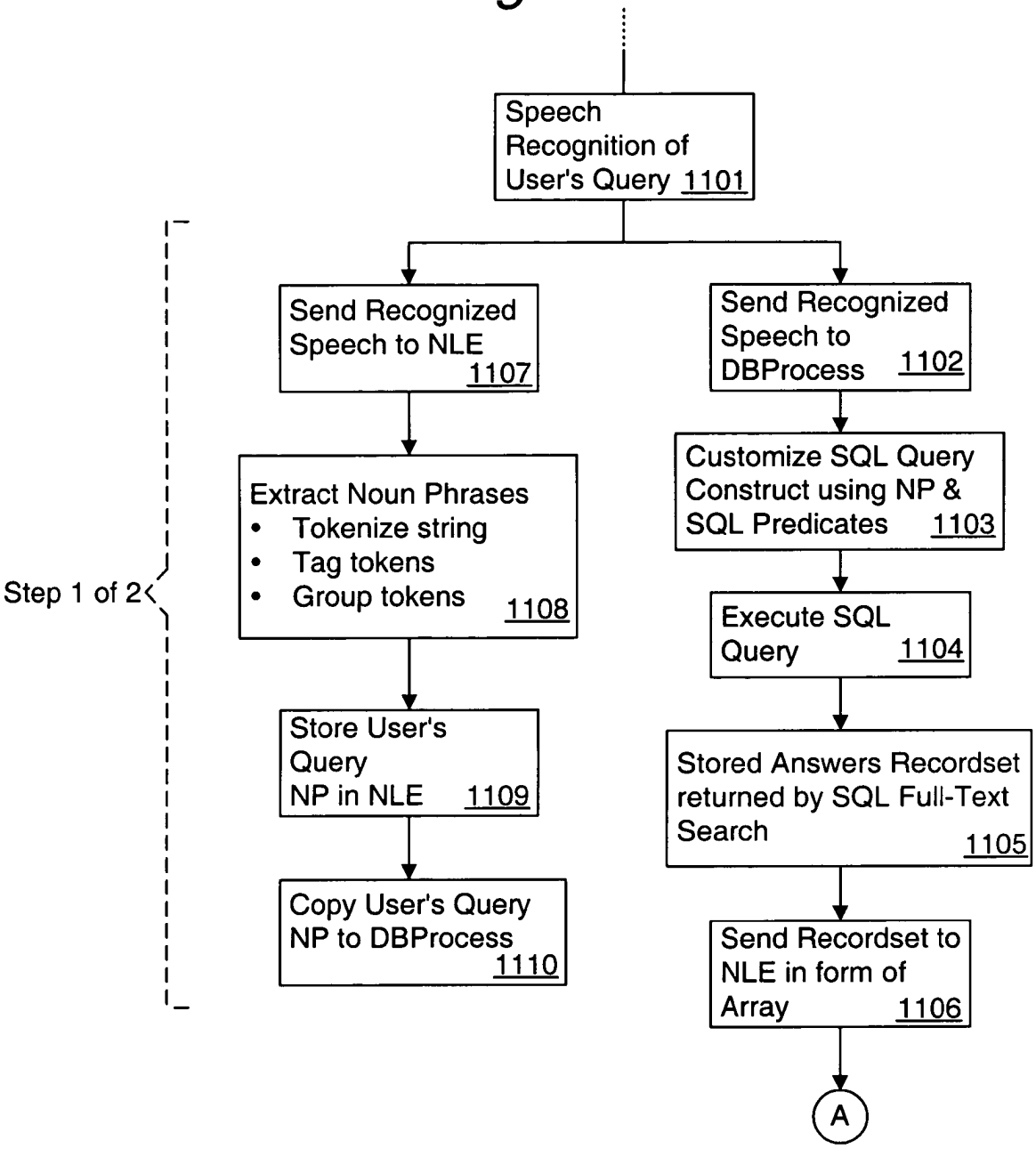

Speech Recognition of User's Query 1101
Send Recognized Speech to NLE 1107
Send Recognized Speech to DBProcess 1102
Extract Noun Phrases
• Tokenize string
• Tag tokens
• Group tokens 1108
Customize SQL Query Construct using NP & SQL Predicates 1103
Execute SQL Query 1104
Store User's Query NP in NLE 1109
Stored Answers Recordset returned by SQL Full-Text Search 1105
Copy User's Query NP to DBProcess 1110
Send Recordset to NLE in form of Array 1106
A
Step 1 of 2

Fig. 11B

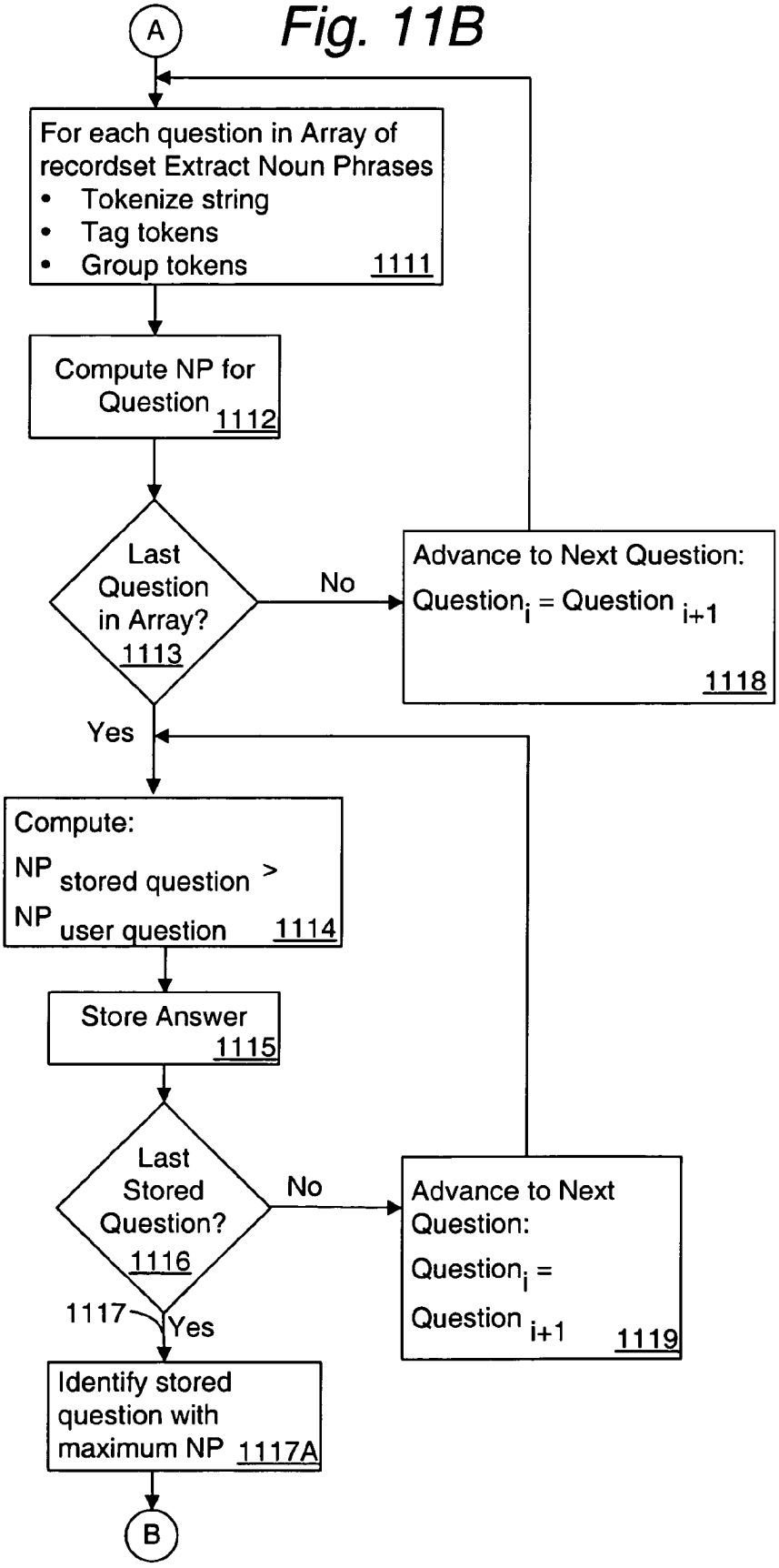

A
For each question in Array of recordset Extract Noun Phrases
• Tokenize string
• Tag tokens
• Group tokens
1111
Compute NP for Question
1112
Last Question in Array?
1113
No
Advance to Next Question:
Question$_i$ = Question $_{i+1}$
1118
Yes
Compute:
NP stored question >
NP user question
1114
Store Answer
1115
Last Stored Question?
1116
No
Advance to Next Question:
Question$_i$ =
Question $_{i+1}$
1119
1117
Yes
Identify stored question with maximum NP
1117A
B

Fig. 11C

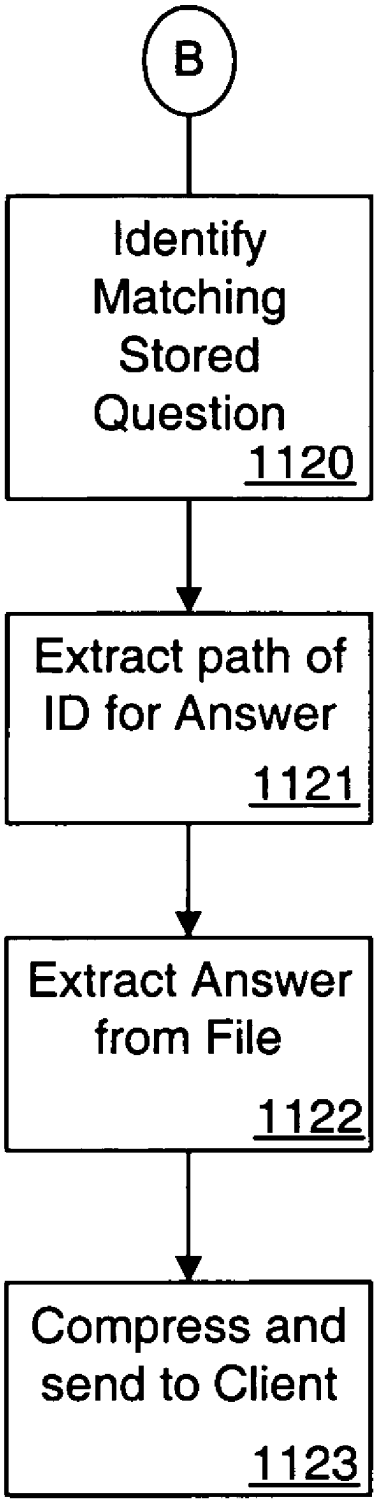

B
Identify Matching Stored Question
1120
Extract path of ID for Answer
1121
Extract Answer from File
1122
Compress and send to Client
1123

Fig. 12

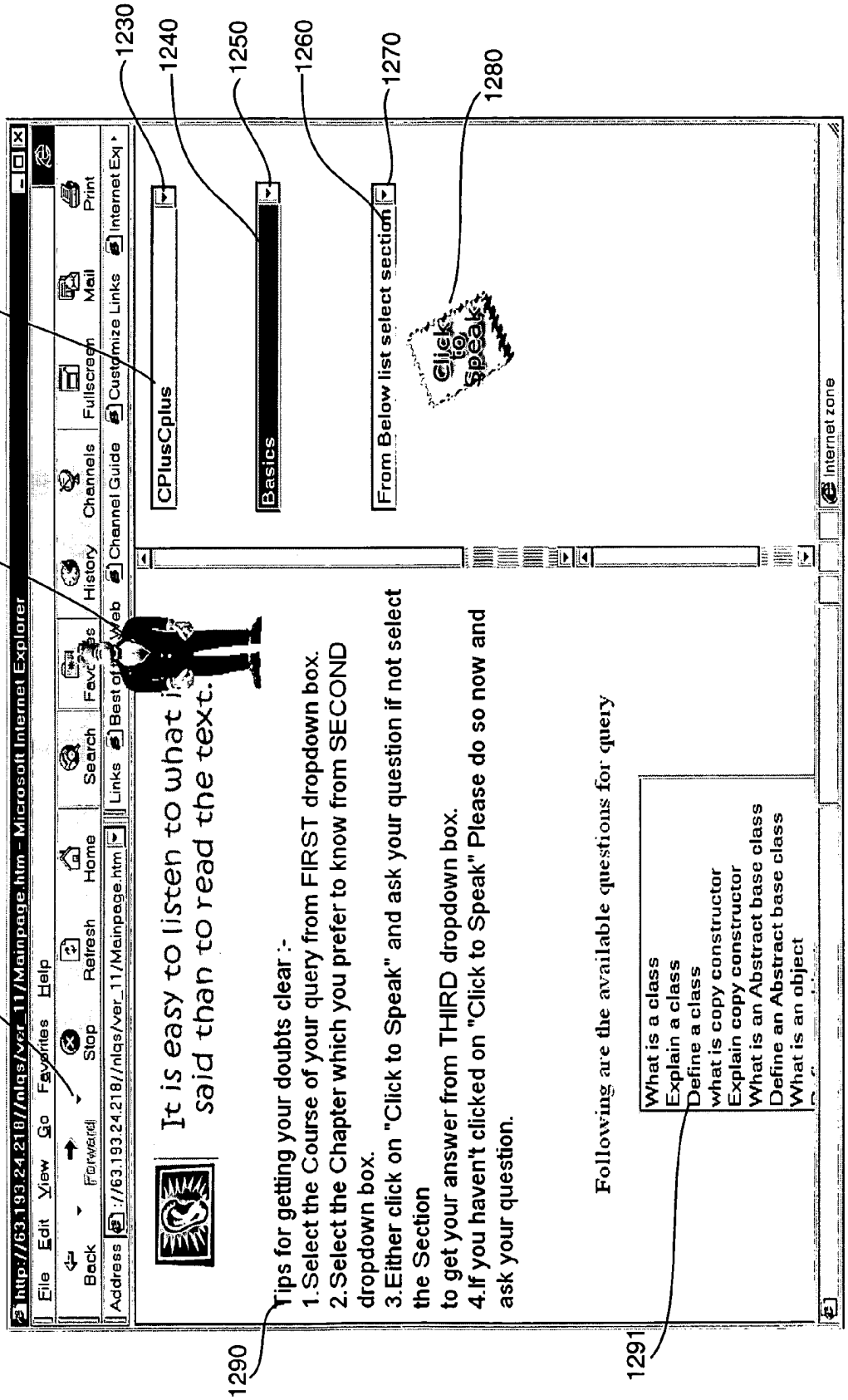

1200
1210
1220
1230
1240
1250
1260
1270
1280
1290
1291
http://63.193.24.218//nlqs/ver_11/Mainpage.htm - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print
Address http://63.193.24.218//nlqs/ver_11/Mainpage.htm
Links Best of the Web Channel Guide Customize Links Internet Exp
CPlusCplus
Basics
From Below list select section
Click to Speak
It is easy to listen to what I said than to read the text.
Tips for getting your doubts clear :-
1.Select the Course of your query from FIRST dropdown box.
2.Select the Chapter which you prefer to know from SECOND dropdown box.
3.Either click on "Click to Speak" and ask your question if not select the Section to get your answer from THIRD dropdown box.
4.If you haven't clicked on "Click to Speak" Please do so now and ask your question.
Following are the available questions for query
What is a class
Explain a class
Define a class
what is copy constructor
Explain copy constructor
What is an Abstract base class
Define an Abstract base class
What is an object
Internet zone

Fig. 13

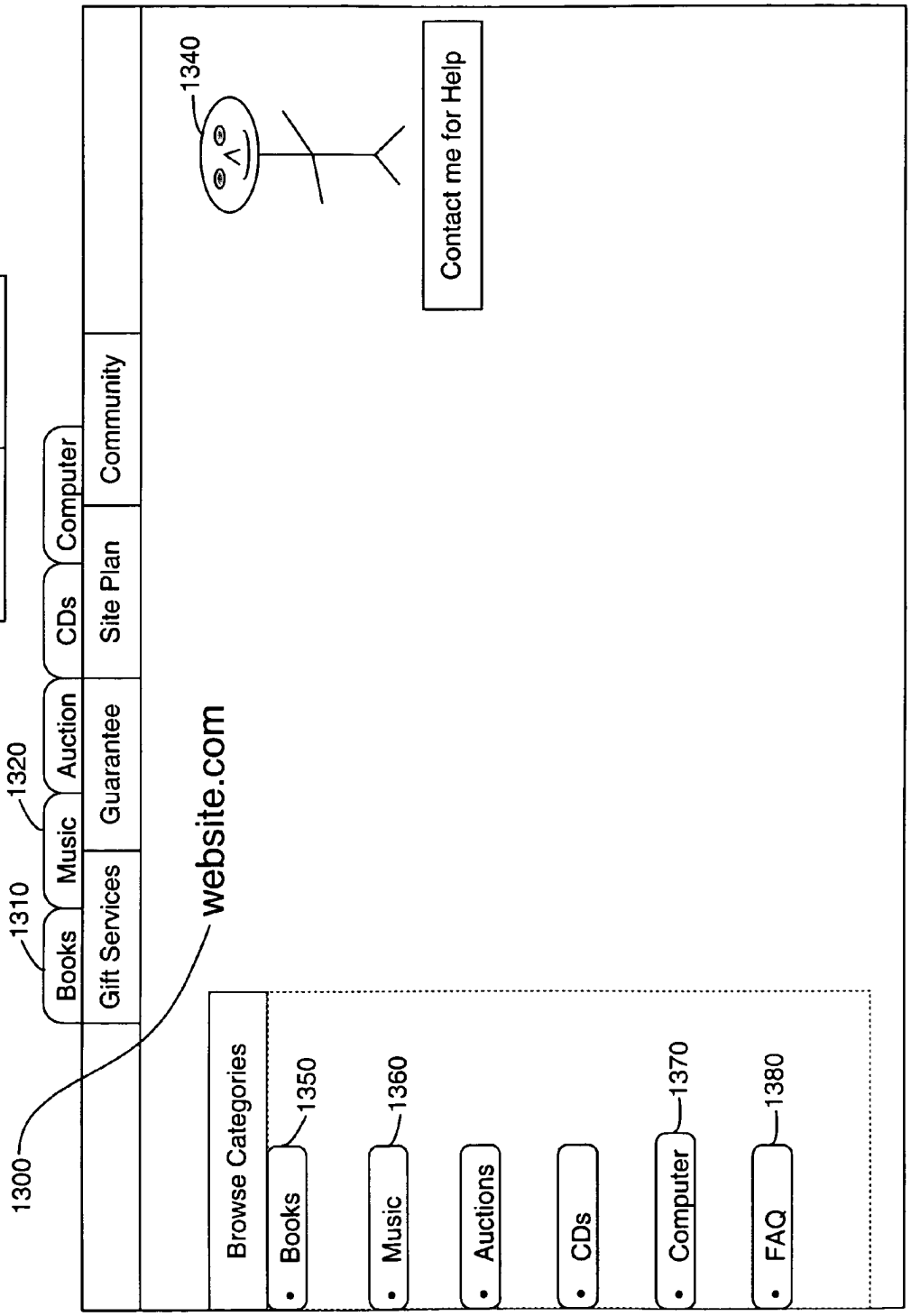

1300
website.com
1310
Books
Gift Services
1320
Music
Guarantee
Auction
CDs
Site Plan
Computer
Community
Account Info
Help
1330
1340
Contact me for Help
Browse Categories
Books
1350
Music
1360
Auctions
CDs
Computer
1370
FAQ
1380

Fig. 14

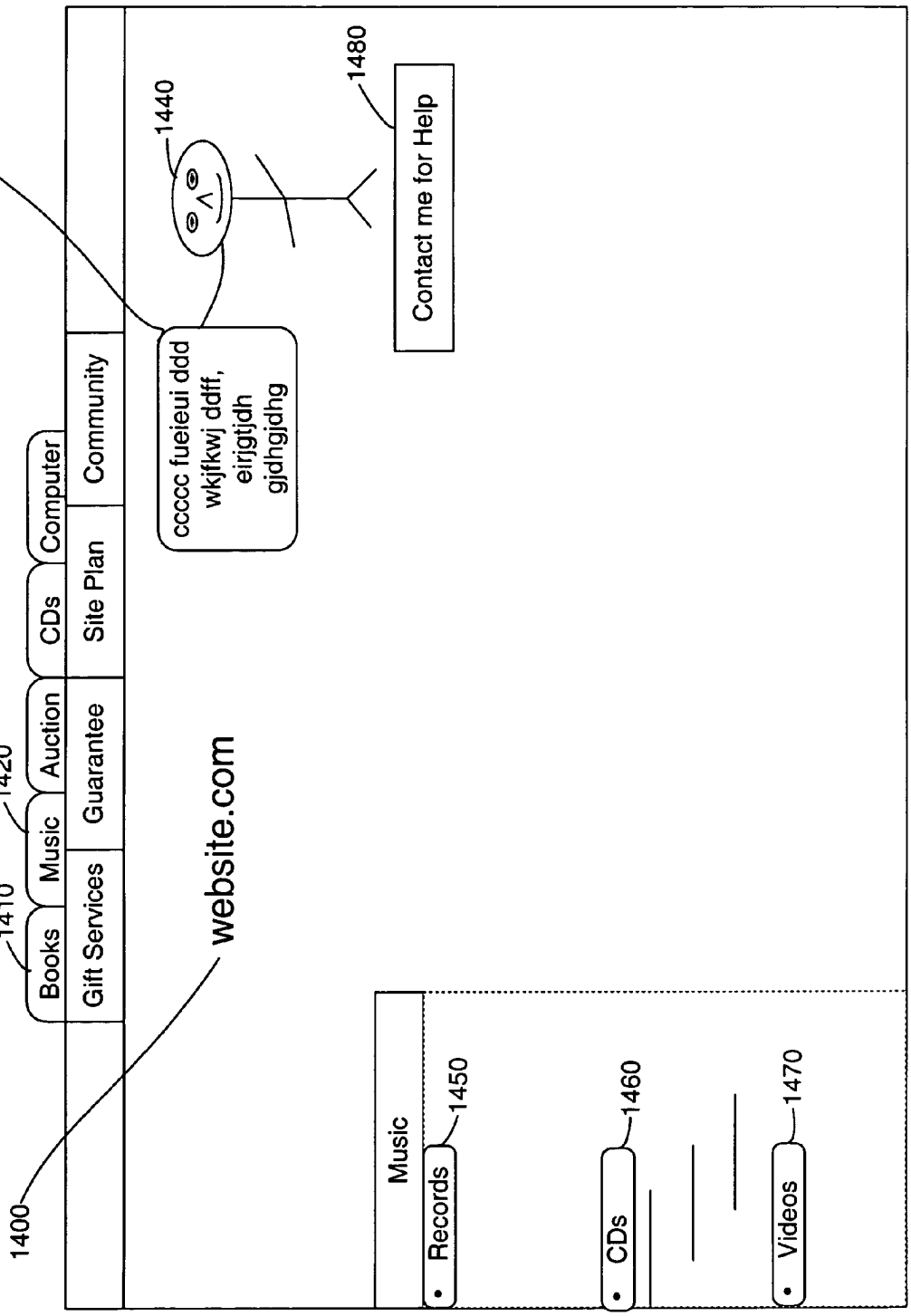

1400
website.com
1410
Books
Gift Services
1420
Music
Guarantee
Auction
CDs
Site Plan
Computer
Community
Account Info
Help
1430
1490
cccc fueieui ddd wkjfkwj ddff, eirjgtjdh gjdhgjdhg
1440
1480
Contact me for Help
Music
Records
1450
CDs
1460
Videos
1470

Fig. 15

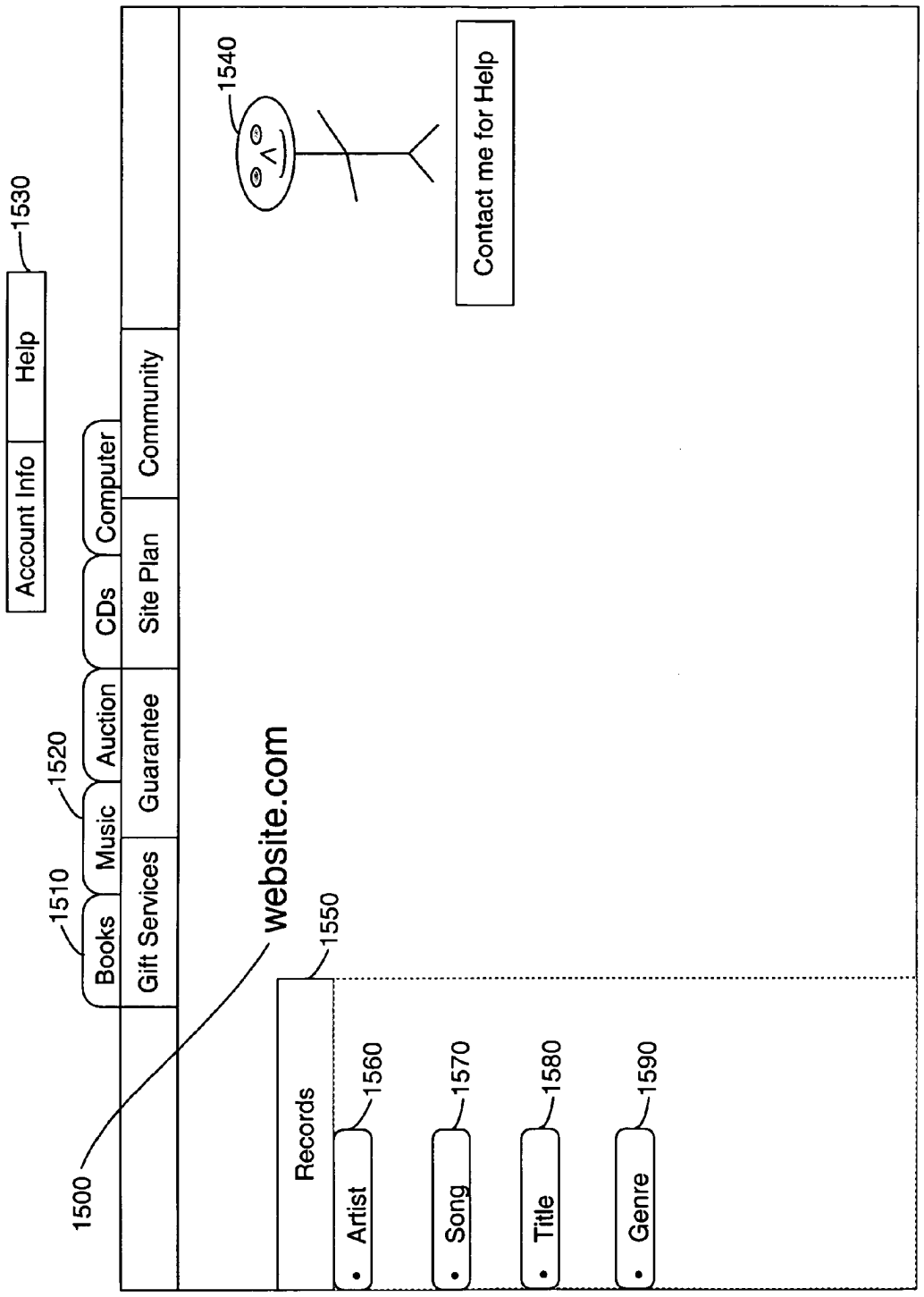

1500
website.com
1510
Books
Music
1520
Auction
Guarantee
Gift Services
CDs
Computer
Site Plan
Community
Account Info
Help
1530
1540
Contact me for Help
1550
Records
Artist
1560
Song
1570
Title
1580
Genre
1590

Fig. 16

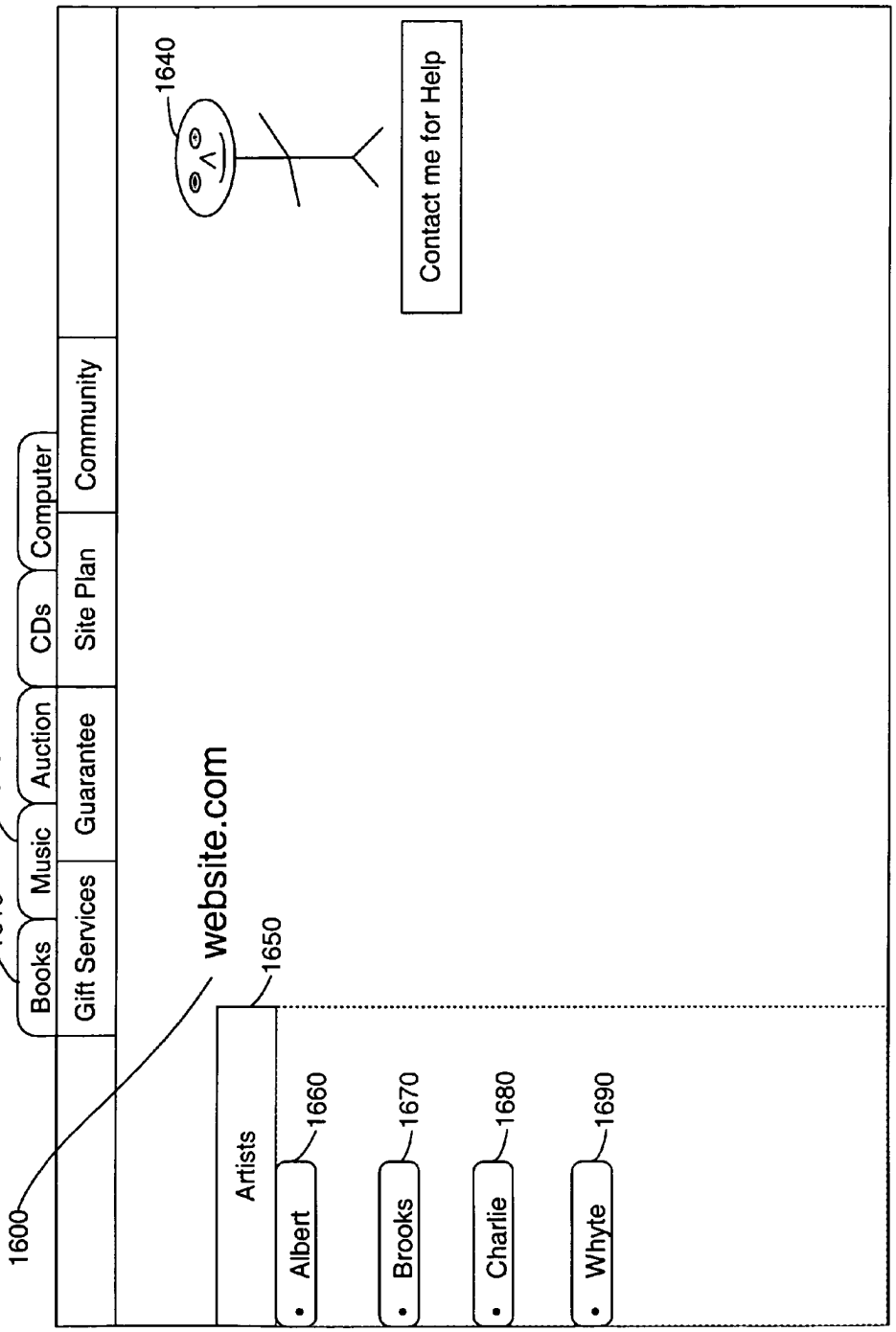

1600
website.com
1610
Books
Music
1620
Auction
Guarantee
Gift Services
CDs
Computer
Site Plan
Community
Account Info
Help
1630
1640
Contact me for Help
1650
Artists
1660
Albert
1670
Brooks
1680
Charlie
1690
Whyte

Fig. 17

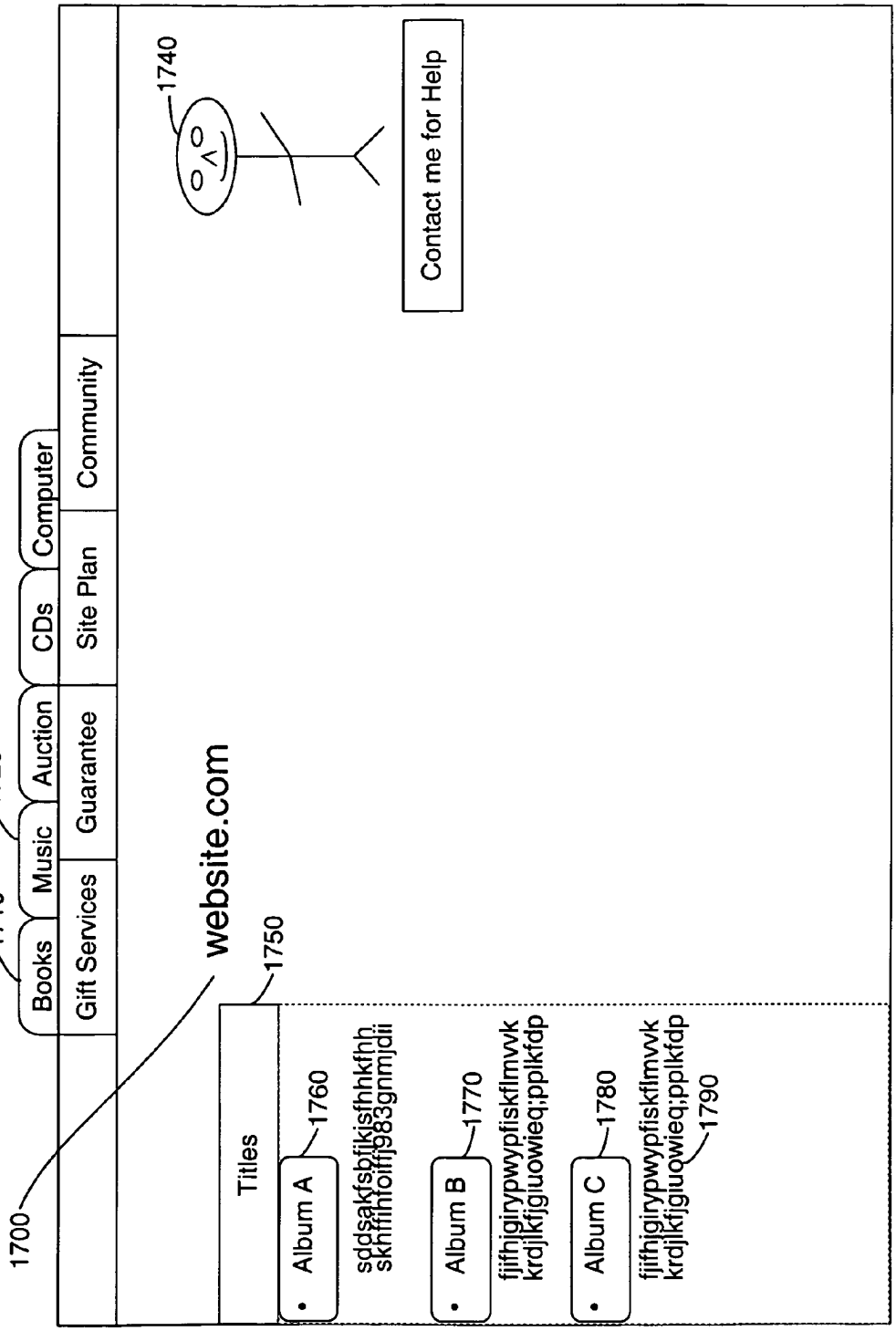

1700
website.com
1710
Books
Music
Gift Services
1720
Auction
Guarantee
CDs
Site Plan
Computer
Community
Account Info
Help
1730
1740
Contact me for Help
1750
Titles
Album A
1760
sddsakfsbfikjsfhhkfhh
skhfhhfoifjf983gnrmjdii
Album B
1770
fjifhjgirypwypfiskflmvvk
krdjlkfjgiuowieq;pplkfdp
Album C
1780
fjifhjgirypwypfiskflmvvk
krdjlkfjgiuowieq;pplkfdp
1790

Fig. 18A

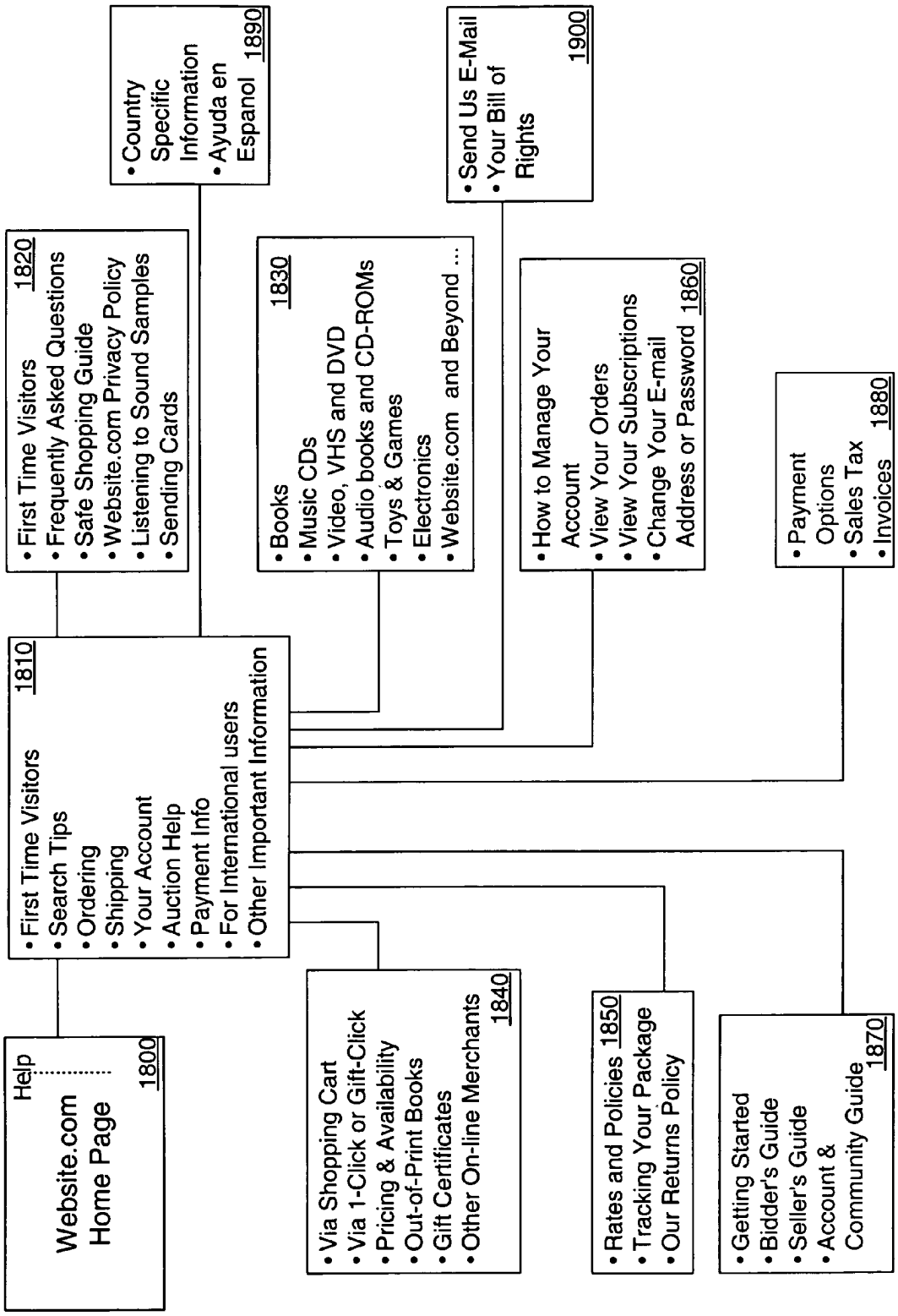
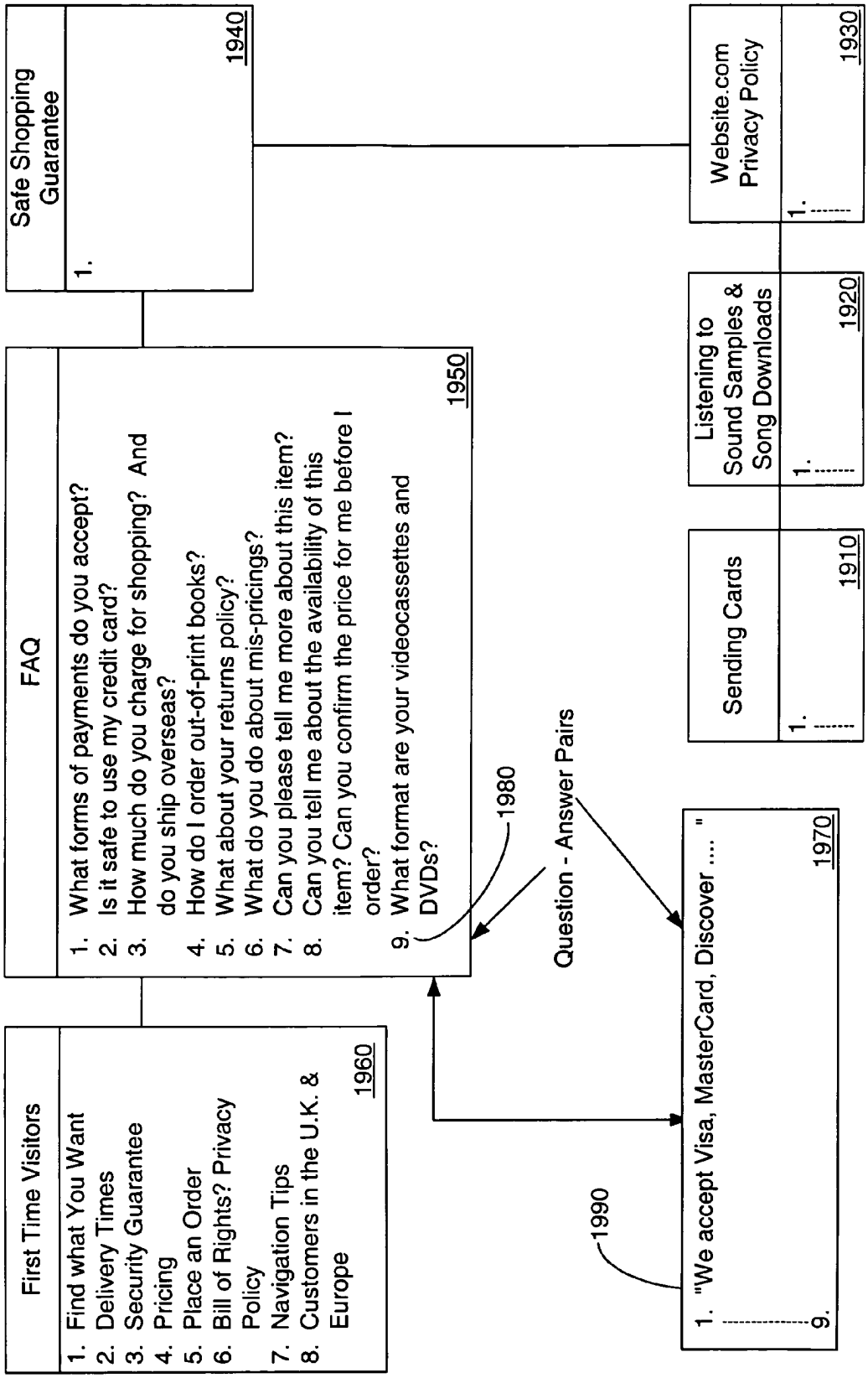

Website.com Home Page 1800
Help
First Time Visitors
Search Tips
Ordering
Shipping
Your Account
Auction Help
Payment Info
For International users
Other Important Information 1810
First Time Visitors 1820
Frequently Asked Questions
Safe Shopping Guide
Website.com Privacy Policy
Listening to Sound Samples
Sending Cards
Country Specific Information
Ayuda en Espanol 1890
Books 1830
Music CDs
Video, VHS and DVD
Audio books and CD-ROMs
Toys & Games
Electronics
Website.com and Beyond ...
Send Us E-Mail
Your Bill of Rights 1900
How to Manage Your Account
View Your Orders
View Your Subscriptions
Change Your E-mail Address or Password 1860
Payment Options
Sales Tax
Invoices 1880
Via Shopping Cart
Via 1-Click or Gift-Click
Pricing & Availability
Out-of-Print Books
Gift Certificates
Other On-line Merchants 1840
Rates and Policies 1850
Tracking Your Package
Our Returns Policy
Getting Started
Bidder's Guide
Seller's Guide
Account & Community Guide 1870

Fig. 18B


Safe Shopping Guarantee
1.
1940
Website.com Privacy Policy
1.
1930
FAQ
1. What forms of payments do you accept?
2. Is it safe to use my credit card?
3. How much do you charge for shopping? And do you ship overseas?
4. How do I order out-of-print books?
5. What about your returns policy?
6. What do you do about mis-pricings?
7. Can you please tell me more about this item?
8. Can you tell me about the availability of this item? Can you confirm the price for me before I order?
9. What format are your videocassettes and DVDs?
1950
Listening to Sound Samples & Song Downloads
1.
1920
First Time Visitors
1. Find what You Want
2. Delivery Times
3. Security Guarantee
4. Pricing
5. Place an Order
6. Bill of Rights? Privacy Policy
7. Navigation Tips
8. Customers in the U.K. & Europe
1960
1980
Question - Answer Pairs
Sending Cards
1.
1910
1990
1. "We accept Visa, MasterCard, Discover .... "
9.
1970

SPEECH RECOGNITION SYSTEM INTERACTIVE AGENT

RELATED APPLICATIONS

The present application claims priority to and is a continuation of Ser. No. 10/684,357 filed Oct. 10, 2003—which in turn is a continuation of Ser. No. 09/439,145 filed Nov. 12, 1999 (now U.S. Pat. No. 6,633,846). Both applications are hereby incorporated by reference herein.

FIELD OF THE INVENTION

The invention relates to a system and an interactive method for responding to speech based user inputs and queries presented over a distributed network such as the INTERNET or local intranet. This interactive system when implemented over the World-Wide Web services (WWW) of the INTERNET, functions so that a client or user can ask a question in a natural language such as English, French, German, Spanish or Japanese and receive the appropriate answer at his or her computer or accessory also in his or her native natural language. The system has particular applicability to such applications as remote learning, e-commerce, technical e-support services, INTERNET searching, etc.

BACKGROUND OF THE INVENTION

The INTERNET, and in particular, the World-Wide Web (WWW), is growing in popularity and usage for both commercial and recreational purposes, and this trend is expected to continue. This phenomenon is being driven, in part, by the increasing and widespread use of personal computer systems and the availability of low cost INTERNET access.

The emergence of inexpensive INTERNET access devices and high speed access techniques such as ADSL, cable modems, satellite modems, and the like, are expected to further accelerate the mass usage of the WWW.

Accordingly, it is expected that the number of entities offering services, products, etc., over the WWW will increase dramatically over the coming years. Until now, however, the INTERNET "experience" for users has been limited mostly to non-voice based input/output devices, such as keyboards, intelligent electronic pads, mice, trackballs, printers, monitors, etc. This presents somewhat of a bottleneck for interacting over the WWW for a variety of reasons.

First, there is the issue of familiarity. Many kinds of applications lend themselves much more naturally and fluently to a voice-based environment. For instance, most people shopping for audio recordings are very comfortable with asking a live sales clerk in a record store for information on titles by a particular author, where they can be found in the store, etc. While it is often possible to browse and search on one's own to locate items of interest, it is usually easier and more efficient to get some form of human assistance first, and, with few exceptions, this request for assistance is presented in the form of a oral query. In addition, many persons cannot or will not, because of physical or psychological barriers, use any of the aforementioned conventional I/O devices. For example, many older persons cannot easily read the text presented on WWW pages, or understand the layout/hierarchy of menus, or manipulate a mouse to make finely coordinated movements to indicate their selections. Many others are intimidated by the look and complexity of computer systems, WWW pages, etc., and therefore do not attempt to use online services for this reason as well.

Thus, applications which can mimic normal human interactions are likely to be preferred by potential on-line shoppers and persons looking for information over the WWW. It is also expected that the use of voice-based systems will increase the universe of persons willing to engage in e-commerce, e-learning, etc. To date, however, there are very few systems, if any, which permit this type of interaction, and, if they do, it is very limited. For example, various commercial programs sold by IBM (VIAVOICE™) and Kurzweil (DRAGON™) permit some user control of the interface (opening, closing files) and searching (by using previously trained URLs) but they do not present a flexible solution that can be used by a number of users across multiple cultures and without time consuming voice training. Typical prior efforts to implement voice based functionality in an INTERNET context can be seen in U.S. Pat. No. 5,819,220 incorporated by reference herein.

Another issue presented by the lack of voice-based systems is efficiency. Many companies are now offering technical support over the INTERNET, and some even offer live operator assistance for such queries. While this is very advantageous (for the reasons mentioned above) it is also extremely costly and inefficient, because a real person must be employed to handle such queries. This presents a practical limit that results in long wait times for responses or high labor overheads. An example of this approach can be seen U.S. Pat. No. 5,802,526 also incorporated by reference herein. In general, a service presented over the WWW is far more desirable if it is "scalable," or, in other words, able to handle an increasing amount of user traffic with little if any perceived delay or troubles by a prospective user.

In a similar context, while remote learning has become an increasingly popular option for many students, it is practically impossible for an instructor to be able to field questions from more than one person at a time. Even then, such interaction usually takes place for only a limited period of time because of other instructor time constraints. To date, however, there is no practical way for students to continue a human-like question and answer type dialog after the learning session is over, or without the presence of the instructor to personally address such queries.

Conversely, another aspect of emulating a human-like dialog involves the use of oral feedback. In other words, many persons prefer to receive answers and information in audible form. While a form of this functionality is used by some websites to communicate information to visitors, it is not performed in a real-time, interactive question-answer dialog fashion so its effectiveness and usefulness is limited.

Yet another area that could benefit from speech-based interaction involves so-called "search" engines used by INTERNET users to locate information of interest at web sites, such as the those available at YAHOO®.com, METACRAWLER®.com, EXCITE®.com, etc. These tools permit the user to form a search query using either combinations of keywords or metacategories to search through a web page database containing text indices associated with one or more distinct web pages. After processing the user's request, therefore, the search engine returns a number of hits which correspond, generally, to URL pointers and text excerpts from the web pages that represent the closest match made by such search engine for the particular user query based on the search processing logic used by search engine. The structure and operation of such prior art search engines, including the mechanism by which they build the web page

database, and parse the search query, are well known in the art. To date, applicant is unaware of any such search engine that can easily and reliably search and retrieve information based on speech input from a user.

There are a number of reasons why the above environments (e-commerce, e-support, remote learning, INTERNET searching, etc.) do not utilize speech-based interfaces, despite the many benefits that would otherwise flow from such capability. First, there is obviously a requirement that the output of the speech recognizer be as accurate as possible. One of the more reliable approaches to speech recognition used at this time is based on the Hidden Markov Model (HMM)—a model used to mathematically describe any time series. A conventional usage of this technique is disclosed, for example, in U.S. Pat. No. 4,587,670 incorporated by reference herein. Because speech is considered to have an underlying sequence of one or more symbols, the HMM models corresponding to each symbol are trained on vectors from the speech waveforms. The Hidden Markov Model is a finite set of states, each of which is associated with a (generally multi-dimensional) probability distribution. Transitions among the states are governed by a set of probabilities called transition probabilities. In a particular state an outcome or observation can be generated, according to the associated probability distribution. This finite state machine changes state once every time unit, and each time t such that a state j is entered, a spectral parameter vector O_t is generated with probability density $B_j(O_t)$. It is only the outcome, not the state visible to an external observer and therefore states are "hidden" to the outside; hence the name Hidden Markov Model. The basic theory of HMMs was published in a series of classic papers by Baum and his colleagues in the late 1960's and early 1970's. HMMs were first used in speech applications by Baker at Carnegie Mellon, by Jelenik and colleagues at IBM in the late 1970's and by Steve Young and colleagues at Cambridge University, UK in the 1990's. Some typical papers and texts are as follows:

1. L. E. Baum, T. Petrie, "Statistical inference for probabilistic functions for finite state Markov chains", Ann. Math. Stat., 37: 1554-1563, 1966
2. L. E. Baum, "An inequality and associated maximation technique in statistical estimation for probabilistic functions of Markov processes", Inequalities 3: 1-8, 1972
3. J. H. Baker, "The dragon system—An Overview", IEEE Trans. on ASSP Proc., ASSP-23(1): 24-29, February 1975
4. F. Jeninek et al, "Continuous Speech Recognition: Statistical methods" in Handbook of Statistics, II, P. R. Kristnaiad, Ed. Amsterdam, The Netherlands, North-Holland, 1982
5. L. R. Bahl, F. Jeninek, R. L. Mercer, "A maximum likelihood approach to continuous speech recognition", IEEE Trans. Pattern Anal. Mach. Intell., PAMI-5: 179-190,1983
6. J. D. Ferguson, "Hidden Markov Analysis: An Introduction", in Hidden Markov Models for Speech, Institute of Defense Analyses, Princeton, N.J. 1980.
7. H. R. Rabiner and B. H. Juang, "Fundamentals of Speech Recognition", Prentice Hall, 1993
8. H. R. Rabiner, "Digital Processing of Speech Signals", Prentice Hall, 1978

More recently research has progressed in extending HMM and combining HMMs with neural networks to speech recognition applications at various laboratories. The following is a representative paper:

9. Nelson Morgan, Hervé Bourlard, Steve Renals, Michael Cohen and Horacio Franco (1993), Hybrid Neural Network/Hidden Markov Model Systems for Continuous Speech Recognition. *Journal of Pattern Recognition and Artificial Intelligence*, Vol. 7, No. 4 pp. 899-916. Also in I. Guyon and P. Wang editors, *Advances in Pattern Recognition Systems using Neural Networks*, Vol. 7 of a Series in Machine Perception and Artificial Intelligence. World Scientific, February 1994.

All of the above are hereby incorporated by reference. While the HMM-based speech recognition yields very good results, contemporary variations of this technique cannot guarantee a word accuracy requirement of 100% exactly and consistently, as will be required for WWW applications for all possible all user and environment conditions. Thus, although speech recognition technology has been available for several years, and has improved significantly, the technical requirements have placed severe restrictions on the specifications for the speech recognition accuracy that is required for an application that combines speech recognition and natural language processing to work satisfactorily.

In contrast to word recognition, Natural language processing (NLP) is concerned with the parsing, understanding and indexing of transcribed utterances and larger linguistic units. Because spontaneous speech contains many surface phenomena such as disfluencies,—hesitations, repairs and restarts, discourse markers such as 'well' and other elements which cannot be handled by the typical speech recognizer, it is the problem and the source of the large gap that separates speech recognition and natural language processing technologies. Except for silence between utterances, another problem is the absence of any marked punctuation available for segmenting the speech input into meaningful units such as utterances. For optimal NLP performance, these types of phenomena should be annotated at its input. However, most continuous speech recognition systems produce only a raw sequence of words. Examples of conventional systems using NLP are shown in U.S. Pat. Nos. 4,991,094, 5,068,789, 5,146,405 and 5,680,628, all of which are incorporated by reference herein.

Second, most of the very reliable voice recognition systems are speaker-dependent, requiring that the interface be "trained" with the user's voice, which takes a lot of time, and is thus very undesirable from the perspective of a WWW environment, where a user may interact only a few times with a particular website. Furthermore, speaker-dependent systems usually require a large user dictionary (one for each unique user) which reduces the speed of recognition. This makes it much harder to implement a real-time dialog interface with satisfactory response capability (i.e., something that mirrors normal conversation—on the order of 3-5 seconds is probably ideal). At present, the typical shrink-wrapped speech recognition application software include offerings from IBM (VIAVOICE™) and Dragon Systems (DRAGON™). While most of these applications are adequate for dictation and other transcribing applications, they are woefully inadequate for applications such as NLQS where the word error rate must be close to 0%. In addition these offerings require long training times and are typically are non client-server configurations. Other types of trained systems are discussed in U.S. Pat. No. 5,231,670 assigned to Kurzweil, and which is also incorporated by reference herein.

Another significant problem faced in a distributed voice-based system is a lack of uniformity/control in the speech recognition process. In a typical stand-alone implementation of a speech recognition system, the entire SR engine runs on

a single client. A well-known system of this type is depicted in U.S. Pat. No. 4,991,217 incorporated by reference herein. These clients can take numerous forms (desktop PC, laptop PC, PDA, etc.) having varying speech signal processing and communications capability. Thus, from the server side perspective, it is not easy to assure uniform treatment of all users accessing a voice-enabled web page, since such users may have significantly disparate word recognition and error rate performances. While a prior art reference to Gould et al.—U.S. Pat. No. 5,915,236—discusses generally the notion of tailoring a recognition process to a set of available computational resources, it does not address or attempt to solve the issue of how to optimize resources in a distributed environment such as a client-server model. Again, to enable such voice-based technologies on a wide-spread scale it is far more preferable to have a system that harmonizes and accounts for discrepancies in individual systems so that even the thinnest client is supportable, and so that all users are able to interact in a satisfactory manner with the remote server running the e-commerce, e-support and/or remote learning application.

Two references that refer to a distributed approach for speech recognition include U.S. Pat. Nos. 5,956,683 and 5,960,399 incorporated by reference herein. In the first of these, U.S. Pat. No. 5,956,683—Distributed Voice Recognition System (assigned to Qualcomm) an implementation of a distributed voice recognition system between a telephony-based handset and a remote station is described. In this implementation, all of the word recognition operations seem to take place at the handset. This is done since the patent describes the benefits that result from locating of the system for acoustic feature extraction at the portable or cellular phone in order to limit degradation of the acoustic features due to quantization distortion resulting from the narrow bandwidth telephony channel. This reference therefore does not address the issue of how to ensure adequate performance for a very thin client platform. Moreover, it is difficult to determine, how, if at all, the system can perform real-time word recognition, and there is no meaningful description of how to integrate the system with a natural language processor.

The second of these references—U.S. Pat. No. 5,960,399—Client/Server Speech Processor/Recognizer (assigned to GTE) describes the implementation of a HMM-based distributed speech recognition system. This reference is not instructive in many respects, however, including how to optimize acoustic feature extraction for a variety of client platforms, such as by performing a partial word recognition process where appropriate. Most importantly, there is only a description of a primitive server-based recognizer that only recognizes the user's speech and simply returns certain keywords such as the user's name and travel destination to fill out a dedicated form on the user's machine. Also, the streaming of the acoustic parameters does not appear to be implemented in real-time as it can only take place after silence is detected. Finally, while the reference mentions the possible use of natural language processing (column 9) there is no explanation of how such function might be implemented in a real-time fashion to provide an interactive feel for the user.

SUMMARY OF THE INVENTION

An object of the present invention, therefore, is to provide an improved system and method for overcoming the limitations of the prior art noted above;

A primary object of the present invention is to provide a word and phrase recognition system that is flexibly and optimally distributed across a client/platform computing architecture, so that improved accuracy, speed and uniformity can be achieved for a wide group of users;

A further object of the present invention is to provide a speech recognition system that efficiently integrates a distributed word recognition system with a natural language processing system, so that both individual words and entire speech utterances can be quickly and accurately recognized in any number of possible languages;

A related object of the present invention is to provide an efficient query response system so that an extremely accurate, real-time set of appropriate answers can be given in response to speech-based queries;

Yet another object of the present invention is to provide an interactive, real-time instructional/learning system that is distributed across a client/server architecture, and permits a real-time question/answer session with an interactive character;

A related object of the present invention is to implement such interactive character with an articulated response capability so that the user experiences a human-like interaction;

Still a further object of the present invention is to provide an INTERNET website with speech processing capability so that voice based data and commands can be used to interact with such site, thus enabling voice-based e-commerce and e-support services to be easily scaleable;

Another object is to implement a distributed speech recognition system that utilizes environmental variables as part of the recognition process to improve accuracy and speed;

A further object is to provide a scaleable query/response database system, to support any number of query topics and users as needed for a particular application and instantaneous demand;

Yet another object of the present invention is to provide a query recognition system that employs a two-step approach, including a relatively rapid first step to narrow down the list of potential responses to a smaller candidate set, and a second more computationally intensive second step to identify the best choice to be returned in response to the query from the candidate set;

A further object of the present invention is to provide a natural language processing system that facilitates query recognition by extracting lexical components of speech utterances, which components can be used for rapidly identifying a candidate set of potential responses appropriate for such speech utterances;

Another related object of the present invention is to provide a natural language processing system that facilitates query recognition by comparing lexical components of speech utterances with a candidate set of potential response to provide an extremely accurate best response to such query.

One general aspect of the present invention, therefore, relates to a natural language query system (NLQS) that offers a fully interactive method for answering user's questions over a distributed network such as the INTERNET or a local intranet. This interactive system when implemented over the worldwide web (WWW) services of the INTERNET functions so that a client or user can ask a question in a natural language such as English, French, German or Spanish and receive the appropriate answer at his or her personal computer also in his or her native natural language.

The system is distributed and consists of a set of integrated software modules at the client's machine and another

set of integrated software programs resident on a server or set of servers. The client-side software program is comprised of a speech recognition program, an agent and its control program, and a communication program. The server-side program is comprised of a communication program, a natural language engine (NLE), a database processor (DBProcess), an interface program for interfacing the DBProcess with the NLE, and a SQL database. In addition, the client's machine is equipped with a microphone and a speaker. Processing of the speech utterance is divided between the client and server side so as to optimize processing and transmission latencies, and so as to provide support for even very thin client platforms.

In the context of an interactive learning application, the system is specifically used to provide a single-best answer to a user's question. The question that is asked at the client's machine is articulated by the speaker and captured by a microphone that is built in as in the case of a notebook computer or is supplied as a standard peripheral attachment. Once the question is captured, the question is processed partially by NLQS client-side software resident in the client's machine. The output of this partial processing is a set of speech vectors that are transported to the server via the INTERNET to complete the recognition of the user's questions. This recognized speech is then converted to text at the server.

After the user's question is decoded by the speech recognition engine (SRE) located at the server, the question is converted to a structured query language (SQL) query. This query is then simultaneously presented to a software process within the server called DBProcess for preliminary processing and to a Natural Language Engine (NLE) module for extracting the noun phrases (NP) of the user's question. During the process of extracting the noun phrase within the NLE, the tokens of the users' question are tagged. The tagged tokens are then grouped so that the NP list can be determined. This information is stored and sent to the DBProcess process.

In the DBProcess, the SQL query is fully customized using the NP extracted from the user's question and other environment variables that are relevant to the application. For example, in a training application, the user's selection of course, chapter and or section would constitute the environment variables. The SQL query is constructed using the extended SQL Full-Text predicates—CONTAINS, FREETEXT, NEAR, AND. The SQL query is next sent to the Full-Text search engine within the SQL database, where a Full-Text search procedure is initiated. The result of this search procedure is recordset of answers. This recordset contains stored questions that are similar linguistically to the user's question. Each of these stored questions has a paired answer stored in a separate text file, whose path is stored in a table of the database.

The entire recordset of returned stored answers is then returned to the NLE engine in the form of an array. Each stored question of the array is then linguistically processed sequentially one by one. This linguistic processing constitutes the second step of a 2-step algorithm to determine the single best answer to the user's question. This second step proceeds as follows: for each stored question that is returned in the recordset, a NP of the stored question is compared with the NP of the user's question. After all stored questions of the array are compared with the user's question, the stored question that yields the maximum match with the user's question is selected as the best possible stored question that matches the user's question. The metric that is used to determine the best possible stored question is the number of noun phrases.

The stored answer that is paired to the best-stored question is selected as the one that answers the user's question. The ID tag of the question is then passed to the DBProcess. This DBProcess returns the answer which is stored in a file.

A communication link is again established to send the answer back to the client in compressed form. The answer once received by the client is decompressed and articulated to the user by the text-to-speech engine. Thus, the invention can be used in any number of different applications involving interactive learning systems, INTERNET related commerce sites, INTERNET search engines, etc.

Computer-assisted instruction environments often require the assistance of mentors or live teachers to answer questions from students. This assistance often takes the form of organizing a separate pre-arranged forum or meeting time that is set aside for chat sessions or live call-in sessions so that at a scheduled time answers to questions may be provided. Because of the time immediacy and the on-demand or asynchronous nature of on-line training where a student may log on and take instruction at any time and at any location, it is important that answers to questions be provided in a timely and cost-effective manner so that the user or student can derive the maximum benefit from the material presented.

This invention addresses the above issues. It provides the user or student with answers to questions that are normally channeled to a live teacher or mentor. This invention provides a single-best answer to questions asked by the student. The student asks the question in his or her own voice in the language of choice. The speech is recognized and the answer to the question is found using a number of technologies including distributed speech recognition, full-text search database processing, natural language processing and text-to-speech technologies. The answer is presented to the user, as in the case of a live teacher, in an articulated manner by an agent that mimics the mentor or teacher, and in the language of choice—English, French, German, Japanese or other natural spoken language. The user can choose the agent's gender as well as several speech parameters such as pitch, volume and speed of the character's voice.

Other applications that benefit from NLQS are e-commerce applications. In this application, the user's query for a price of a book, compact disk or for the availability of any item that is to be purchased can be retrieved without the need to pick through various lists on successive web pages. Instead, the answer is provided directly to the user without any additional user input.

Similarly, it is envisioned that this system can be used to provide answers to frequently-asked questions (FAQs), and as a diagnostic service tool for e-support. These questions are typical of a give web site and are provided to help the user find information related to a payment procedure or the specifications of, or problems experienced with a product/service. In all of these applications, the NLQS architecture can be applied.

A number of inventive methods associated with these architectures are also beneficially used in a variety of INTERNET related applications.

Although the inventions are described below in a set of preferred embodiments, it will be apparent to those skilled in the art the present inventions could be beneficially used in many environments where it is necessary to implement fast, accurate speech recognition, and/or to provide a human-like dialog capability to an intelligent system.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram of a preferred embodiment of a natural language query system (NLQS) of the present invention, which is distributed across a client/server computing architecture, and can be used as an interactive learning system, an e-commerce system, an e-support system, and the like;

FIGS. **2**A-**2**C are a block diagram of a preferred embodiment of a client side system, including speech capturing modules, partial speech processing modules, encoding modules, transmission modules, agent control modules, and answer/voice feedback modules that can be used in the aforementioned NLQS;

FIG. **2**D is a block diagram of a preferred embodiment of a set of initialization routines and procedures used for the client side system of FIGS. **2**A-**2**C;

FIG. **3** is a block diagram of a preferred embodiment of a set of routines and procedures used for handling an iterated set of speech utterances on the client side system of FIGS. **2**A-**2**C, transmitting speech data for such utterances to a remote server, and receiving appropriate responses back from such server;

FIG. **4** is a block diagram of a preferred embodiment of a set of initialization routines and procedures used for un-initializing the client side system of FIGS. **2**A-**2**C;

FIG. **4**A is a block diagram of a preferred embodiment of a set of routines and procedures used for implementing a distributed component of a speech recognition module for the server side system of FIG. **5**;

FIG. **4**B is a block diagram of a preferred set of routines and procedures used for implementing an SQL query builder for the server side system of FIG. **5**;

FIG. **4**C is a block diagram of a preferred embodiment of a set of routines and procedures used for implementing a database control process module for the server side system of FIG. **5**;

FIG. **4**D is a block diagram of a preferred embodiment of a set of routines and procedures used for implementing a natural language engine that provides query formulation support, a query response module, and an interface to the database control process module for the server side system of FIG. **5**;

FIG. **5** is a block diagram of a preferred embodiment of a server side system, including a speech recognition module to complete processing of the speech utterances, environmental and grammar control modules, query formulation modules, a natural language engine, a database control module, and a query response module that can be used in the aforementioned NLQS;

FIG. **6** illustrates the organization of a full-text database used as part of server side system shown in FIG. **5**;

FIG. **7**A illustrates the organization of a full-text database course table used as part of server side system shown in FIG. **5** for an interactive learning embodiment of the present invention;

FIG. **7**B illustrates the organization of a full-text database chapter table used as part of server side system shown in FIG. **5** for an interactive learning embodiment of the present invention;

FIG. **7**C describes the fields used in a chapter table used as part of server side system shown in FIG. **5** for an interactive learning embodiment of the present invention;

FIG. **7**D describes the fields used in a section table used as part of server side system shown in FIG. **5** for an interactive learning embodiment of the present invention;

FIG. **8** is a flow diagram of a first set of operations performed by a preferred embodiment of a natural language engine on a speech utterance including Tokenization, Tagging and Grouping;

FIG. **9** is a flow diagram of the operations performed by a preferred embodiment of a natural language engine on a speech utterance including stemming and Lexical Analysis

FIG. **10** is a block diagram of a preferred embodiment of a SQL database search and support system for the present invention;

FIGS. **11**A-**11**C are flow diagrams illustrating steps performed in a preferred two step process implemented for query recognition by the NLQS of FIG. **2**;

FIG. **12** is an illustration of another embodiment of the present invention implemented as part of a Web-based speech based learning/training System;

FIGS. **13-17** are illustrations of another embodiment of the present invention implemented as part of a Web-based e-commerce system;

FIG. **18** is an illustration of another embodiment of the present invention implemented as part of a voice-based Help Page for an E-Commerce Web Site.

DETAILED DESCRIPTION OF THE INVENTION

Overview

As alluded to above, the present inventions allow a user to ask a question in a natural language such as English, French, German, Spanish or Japanese at a client computing system (which can be as simple as a personal digital assistant or cell-phone, or as sophisticated as a high end desktop PC) and receive an appropriate answer from a remote server also in his or her native natural language. As such, the embodiment of the invention shown in FIG. **1** is beneficially used in what can be generally described as a Natural Language Query System (NLQS) **100**, which is configured to interact on a real-time basis to give a human-like dialog capability/experience for e-commerce, e-support, and e-learning applications.

The processing for NLQS **100** is generally distributed across a client side system **150**, a data link **160**, and a server-side system **180**. These components are well known in the art, and in a preferred embodiment include a personal computer system **150**, an INTERNET connection **160**A, **160**B, and a larger scale computing system **180**. It will be understood by those skilled in the art that these are merely exemplary components, and that the present invention is by no means limited to any particular implementation or combination of such systems. For example, client-side system **150** could also be implemented as a computer peripheral, a PDA, as part of a cell-phone, as part of an INTERNET-adapted appliance, an INTERNET linked kiosk, etc. Similarly, while an INTERNET connection is depicted for data link **160**A, it is apparent that any channel that is suitable for carrying data between client system **150** and server system **180** will suffice, including a wireless link, an RF link, an IR link, a LAN, and the like. Finally, it will be further appreciated that server system **180** may be a single, large-scale system, or a collection of smaller systems interlinked to support a number of potential network users.

Initially speech input is provided in the form of a question or query articulated by the speaker at the client's machine or personal accessory as a speech utterance. This speech utterance is captured and partially processed by NLQS client-side software **155** resident in the client's machine. To

facilitate and enhance the human-like aspects of the interaction, the question is presented in the presence of an animated character **157** visible to the user who assists the user as a personal information retriever/agent. The agent can also interact with the user using both visible text output on a monitor/display (not shown) and/or in audible form using a text to speech engine **159**. The output of the partial processing done by SRE **155** is a set of speech vectors that are transmitted over communication channel **160** that links the user's machine or personal accessory to a server or servers via the INTERNET or a wireless gateway that is linked to the INTERNET as explained above. At server **180**, the partially processed speech signal data is handled by a server-side SRE **182**, which then outputs recognized speech text corresponding to the user's question. Based on this user question related text, a text-to-query converter **184** formulates a suitable query that is used as input to a database processor **186**. Based on the query, database processor **186** then locates and retrieves an appropriate answer using a customized SQL query from database **188**. A Natural Language Engine **190** facilitates structuring the query to database **188**. After a matching answer to the user's question is found, the former is transmitted in text form across data link **160**B, where it is converted into speech by text to speech engine **159**, and thus expressed as oral feedback by animated character agent **157**.

Because the speech processing is broken up in this fashion, it is possible to achieve real-time, interactive, human-like dialog consisting of a large, controllable set of questions/answers. The assistance of the animated agent **157** further enhances the experience, making it more natural and comfortable for even novice users. To make the speech recognition process more reliable, context-specific grammars and dictionaries are used, as well as natural language processing routines at NLE **190**, to analyze user questions lexically. While context-specific processing of speech data is known in the art (see e.g., U.S. Pat. Nos. 5,960,394, 5,867,817, 5,758,322 and 5,384,892 incorporated by reference herein) the present inventors are unaware of any such implementation as embodied in the present inventions. The text of the user's question is compared against text of other questions to identify the question posed by the user by DB processor/engine (DBE) **186**. By optimizing the interaction and relationship of the SR engines **155** and **182**, the NLP routines **190**, and the dictionaries and grammars, an extremely fast and accurate match can be made, so that a unique and responsive answer can be provided to the user.

On the server side **180**, interleaved processing further accelerates the speech recognition process. In simplified terms, the query is presented simultaneously both to NLE **190** after the query is formulated, as well as to DBE **186**. NLE **190** and SRE **182** perform complementary functions in the overall recognition process. In general, SRE **182** is primarily responsible for determining the identity of the words articulated by the user, while NLE **190** is responsible for the linguistic morphological analysis of both the user's query and the search results returned after the database query.

After the user's query is analyzed by NLE **190** some parameters are extracted and sent to the DBProcess. Additional statistics are stored in an array for the 2^{nd} step of processing. During the 2^{nd} step of 2-step algorithm, the recordset of preliminary search results are sent to the NLE **160** for processing. At the end of this 2^{nd} step, the single question that matches the user's query is sent to the DBProcess where further processing yields the paired answer that is paired with the single best stored question.

Thus, the present invention uses a form of natural language processing (NLP) to achieve optimal performance in a speech based web application system. While NLP is known in the art, prior efforts in Natural Language Processing (NLP) work nonetheless have not been well integrated with Speech Recognition (SR) technologies to achieve reasonable results in a web-based application environment. In speech recognition, the result is typically a lattice of possible recognized words each with some probability of fit with the speech recognizer. As described before, the input to a typical NLP system is typically a large linguistic unit. The NLP system is then charged with the parsing, understanding and indexing of this large linguistic unit or set of transcribed utterances. The result of this NLP process is to understand lexically or morphologically the entire linguistic unit as opposed to word recognition. Put another way, the linguistic unit or sentence of connected words output by the SRE has to be understood lexically, as opposed to just being "recognized".

As indicated earlier, although speech recognition technology has been available for several years, the technical requirements for the NLQS invention have placed severe restrictions on the specifications for the speech recognition accuracy that is required for an application that combines speech recognition and natural language processing to work satisfactorily. In realizing that even with the best of conditions, it might be not be possible to achieve the perfect 100% speech recognition accuracy that is required, the present invention employs an algorithm that balances the potential risk of the speech recognition process with the requirements of the natural language processing so that even in cases where perfect speech recognition accuracy is not achieved for each word in the query, the entire query itself is nonetheless recognized with sufficient accuracy.

This recognition accuracy is achieved even while meeting very stringent user constraints, such as short latency periods of 3 to 5 seconds (ideally—ignoring transmission latencies which can vary) for responding to a speech-based query, and for a potential set of 100-250 query questions. This quick response time gives the overall appearance and experience of a real-time discourse that is more natural and pleasant from the user's perspective. Of course, non-real time applications, such as translation services for example, can also benefit from the present teachings as well, since a centralized set of HMMs, grammars, dictionaries, etc., are maintained.

General Aspects of Speech Recognition used in the Present Inventions

General background information on speech recognition can be found in the prior art references discussed above and incorporated by reference herein. Nonetheless, a discussion of some particular exemplary forms of speech recognition structures and techniques that are well-suited for NLQS **100** is provided next to better illustrate some of the characteristics, qualities and features of the present inventions.

Speech recognition technology is typically of two types—speaker independent and speaker dependent. In speaker-dependent speech recognition technology, each user has a voice file in which a sample of each potentially recognized word is stored. Speaker-dependent speech recognition systems typically have large vocabularies and dictionaries making them suitable for applications as dictation and text transcribing. It follows also that the memory and processor resource requirements for the speaker-dependent can be and are typically large and intensive.

Conversely speaker-independent speech recognition technology allows a large group of users to use a single vocabulary file. It follows then that the degree of accuracy that can be achieved is a function of the size and complexity of the grammars and dictionaries that can be supported for a given language. Given the context of applications for which NLQS, the use of small grammars and dictionaries allow speaker independent speech recognition technology to be implemented in NLQS.

The key issues or requirements for either type—speaker-independent or speaker-dependent, are accuracy and speed. As the size of the user dictionaries increase, the speech recognition accuracy metric—word error rate (WER) and the speed of recognition decreases. This is so because the search time increases and the pronunciation match becomes more complex as the size of the dictionary increases.

The basis of the NLQS speech recognition system is a series of Hidden Markov Models (HMM), which, as alluded to earlier, are mathematical models used to characterize any time varying signal. Because parts of speech are considered to be based on an underlying sequence of one or more symbols, the HMM models corresponding to each symbol are trained on vectors from the speech waveforms. The Hidden Markov Model is a finite set of states, each of which is associated with a (generally multi-dimensional) probability distribution. Transitions among the states are governed by a set of probabilities called transition probabilities. In a particular state an outcome or observation can be generated, according to an associated probability distribution. This finite state machine changes state once every time unit, and each time t such that a state j is entered, a spectral parameter vector O_t is generated with probability density $B_j(O_t)$. It is only the outcome, not the state which is visible to an external observer and therefore states are "hidden" to the outside; hence the name Hidden Markov Model.

In isolated speech recognition, it is assumed that the sequence of observed speech vectors corresponding to each word can each be described by a Markov model as follows:

$$O=o_1, o_2, \ldots o_T \quad (1\text{-}1)$$

where o_t is a speech vector observed at time t. The isolated word recognition then is to compute:

$$\arg\max \{P(w_i|O)\} \quad (1\text{-}2)$$

By using Bayes' Rule,

$$\{P(w_i|O)\}=[P(O|w_i)P(w_i)]/P(O) \quad (1\text{-}3)$$

In the general case, the Markov model when applied to speech also assumes a finite state machine which changes state once every time unit and each time that a state j is entered, a speech vector o_t is generated from the probability density $b_j(o_t)$. Furthermore, the transition from state i to state j is also probabilistic and is governed by the discrete probability a_{ij}.

For a state sequence X, the joint probability that O is generated by the model M moving through a state sequence X is the product of the transition probabilities and the output probabilities. Only the observation sequence is known—the state sequence is hidden as mentioned before.

Given that X is unknown, the required likelihood is computed by summing over all possible state sequences X=x(1), x(2), x(3) . . . x(T), that is

$$P(O|M)=\Sigma\{a_{x(0)x(1)}\Pi b(x)(o_t)a_{x(t)x(t+1)}\}$$

Given a set of models M_i, corresponding to words w_i equation 1-2 is solved by using 1-3 and also by assuming that:

$$P(O|w_i)=P(O|M_i)$$

All of this assumes that the parameters $\{a_{ij}\}$ and $\{b_j(o_t)\}$ are known for each model M_i. This can be done, as explained earlier, by using a set of training examples corresponding to a particular model. Thereafter, the parameters of that model can be determined automatically by a robust and efficient re-estimation procedure. So if a sufficient number of representative examples of each word are collected, then a HMM can be constructed which simply models all of the many sources of variability inherent in real speech. This training is well-known in the art, so it is not described at length herein, except to note that the distributed architecture of the present invention enhances the quality of HMMs, since they are derived and constituted at the server side, rather than the client side. In this way, appropriate samples from users of different geographical areas can be easily compiled and analyzed to optimize the possible variations expected to be seen across a particular language to be recognized. Uniformity of the speech recognition process is also well-maintained, and error diagnostics are simplified, since each prospective user is using the same set of HMMs during the recognition process.

To determine the parameters of a HMM from a set of training samples, the first step typically is to make a rough guess as to what they might be. Then a refinement is done using the Baum-Welch estimation formulae. By these formulae, the maximum likelihood estimates of μ_j (where μ_j is mean vector and Σ_j is covariance matrix) is:

$$\mu_j=\Sigma^T_{t=1}L_j(t)o_t/[\Sigma^T_{t=1}L_j(t)o_t]$$

A forward-backward algorithm is next used to calculate the probability of state occupation $L_j(t)$. If the forward probability $\alpha_j(t)$ for some model M with N states is defined as:

$$\alpha_j(t)=P(o_1, \ldots, o_t, x(t)=j|M)$$

This probability can be calculated using the recursion:

$$\alpha_j(t)=[\Sigma^{N-1}_{i=2}\alpha(t-1)a_{ij}]b_j(o_t)$$

Similarly the backward probability can be computed using the recursion:

$$\beta_j(t)=\Sigma^{N-1}_{j=2}a_{ij}b_j(o_{t+1})(t+1)$$

Realizing that the forward probability is a joint probability and the backward probability is a conditional probability, the probability of state occupation is the product of the two probabilities:

$$\alpha j(t)\beta_j(t)=P(O,x(t)=j|M)$$

Hence the probability of being in state j at a time t is:

$$L_j(t)=1/P[\alpha_j(t)\beta_j(t)]$$

where P=P(O|M)

To generalize the above for continuous speech recognition, we assume the maximum likelihood state sequence where the summation is replaced by a maximum operation. Thus for a given model M, let $\phi j(t)$ represent the maximum likelihood of observing speech vectors o_1 to o_t and being used in state j at time t:

$$\phi_j(t)=\max\{\phi j(t)(t-1)\alpha_{ij}\}\beta_j(o_t)$$

Expressing this logarithmically to avoid underflow, this likelihood becomes:

$$\psi_j(t)=\max\{\psi_i(t-1)+\log(\alpha_{ij})\}+\log(b_j(o_t)$$

This is also known as the Viterbi algorithm. It can be visualized as finding the best path through a matrix where the vertical dimension represents the states of the HMM and

horizontal dimension represents frames of speech i.e. time. To complete the extension to connected speech recognition, it is further assumed that each HMM representing the underlying sequence is connected. Thus the training data for continuous speech recognition should consist of connected utterances; however, the boundaries between words do not have to be known.

To improve computational speed/efficiency, the Viterbi algorithm is sometimes extended to achieve convergence by using what is known as a Token Passing Model. The token passing model represents a partial match between the observation sequence o_1 to o_t and a particular model, subject to the constraint that the model is in state j at time t. This token passing model can be extended easily to connected speech environments as well if we allow the sequence of HMMs to be defined as a finite state network. A composite network that includes both phoneme-based HMMs and complete words can be constructed so that a single-best word can be recognized to form connected speech using word N-best extraction from the lattice of possibilities. This composite form of HMM-based connected speech recognizer is the basis of the NLQS speech recognizer module. Nonetheless, the present invention is not limited as such to such specific forms of speech recognizers, and can employ other techniques for speech recognition if they are otherwise compatible with the present architecture and meet necessary performance criteria for accuracy and speed to provide a real-time dialog experience for users.

The representation of speech for the present invention's HMM-based speech recognition system assumes that speech is essentially either a quasi-periodic pulse train (for voiced speech sounds) or a random noise source (for unvoiced sounds). It may be modeled as two sources—one a impulse train generator with pitch period P and a random noise generator which is controlled by a voice/unvoiced switch. The output of the switch is then fed into a gain function estimated from the speech signal and scaled to feed a digital filter H(z) controlled by the vocal tract parameter characteristics of the speech being produced. All of the parameters for this model—the voiced/unvoiced switching, the pitch period for voiced sounds, the gain parameter for the speech signal and the coefficient of the digital filter, vary slowly with time. In extracting the acoustic parameters from the user's speech input so that it can evaluated in light of a set of HMMs, cepstral analysis is typically used to separate the vocal tract information from the excitation information. The cepstrum of a signal is computed by taking the Fourier (or similar) transform of the log spectrum. The principal advantage of extracting cepstral coefficients is that they are de-correlated and the diagonal covariances can be used with HMMs. Since the human ear resolves frequencies non-linearly across the audio spectrum, it has been shown that a front-end that operates in a similar non-linear way improves speech recognition performance.

Accordingly, instead of a typical linear prediction-based analysis, the front-end of the NLQS speech recognition engine implements a simple, fast Fourier transform based filter bank designed to give approximately equal resolution on the Mel-scale. To implement this filter bank, a window of speech data (for a particular time frame) is transformed using a software based Fourier transform and the magnitude taken. Each FFT magnitude is then multiplied by the corresponding filter gain and the results accumulated. The cepstral coefficients that are derived from this filter-bank analysis at the front end are calculated during a first partial processing phase of the speech signal by using a Discrete Cosine Transform of the log filter bank amplitudes. These cepstral coefficients are called Mel-Frequency Cepstral Coefficients (MFCC) and they represent some of the speech parameters transferred from the client side to characterize the acoustic features of the user's speech signal. These parameters are chosen for a number of reasons, including the fact that they can be quickly and consistently derived even across systems of disparate capabilities (i.e., for everything from a low power PDA to a high powered desktop system), they give good discrimination, they lend themselves to a number of useful recognition related manipulations, and they are relatively small and compact in size so that they can be transported rapidly across even a relatively narrow band link. Thus, these parameters represent the least amount of information that can be used by a subsequent server side system to adequately and quickly complete the recognition process.

To augment the speech parameters an energy term in the form of the logarithm of the signal energy is added. Accordingly, RMS energy is added to the 12 MFCC's to make **13** coefficients. These coefficients together make up the partially processed speech data transmitted in compressed form from the user's client system to the remote server side.

The performance of the present speech recognition system is enhanced significantly by computing and adding time derivatives to the basic static MFCC parameters at the server side. These two other sets of coefficients—the delta and acceleration coefficients representing change in each of the 13 values from frame to frame (actually measured across several frames), are computed during a second partial speech signal processing phase to complete the initial processing of the speech signal, and are added to the original set of coefficients after the latter are received. These MFCCs together with the delta and acceleration coefficients constitute the observation vector O_t mentioned above that is used for determining the appropriate HMM for the speech data.

The delta and acceleration coefficients are computed using the following regression formula:

$$d_t=\Sigma^{\theta}_{\theta=1}[c_{t+\theta}-c_{t-\theta}]/2\Sigma^{\theta}_{\theta=1}\theta^2$$

where d_t is a delta coefficient at time t computed in terms of the corresponding static coefficients:

$$d_t=[c_{t+\theta}-c_{t-\theta}]/2\theta$$

In a typical stand-alone implementation of a speech recognition system, the entire SR engine runs on a single client. In other words, both the first and second partial processing phases above are executed by the same DSP (or microprocessor) running a ROM or software code routine at the client's computing machine.

In contrast, because of several considerations, specifically—cost, technical performance, and client hardware uniformity, the present NLQS system uses a partitioned or distributed approach. While some processing occurs on the client side, the main speech recognition engine runs on a centrally located server or number of servers. More specifically, as noted earlier, capture of the speech signals, MFCC vector extraction and compression are implemented on the client's machine during a first partial processing phase. The routine is thus streamlined and simple enough to be implemented within a browser program (as a plug in module, or a downloadable applet for example) for maximum ease of use and utility. Accordingly, even very "thin" client platforms can be supported, which enables the use of the present system across a greater number of potential sites. The primary MFCCs are then transmitted to the server over the channel, which, for example, can include a dial-up INTER-

NET connection, a LAN connection, a wireless connection and the like. After decompression, the delta and acceleration coefficients are computed at the server to complete the initial speech processing phase, and the resulting observation vectors O_t are also determined.

General Aspects of Speech Recognition Engine

The speech recognition engine is also located on the server, and is based on a HTK-based recognition network compiled from a word-level network, a dictionary and a set of HMMs. The recognition network consists of a set of nodes connected by arcs. Each node is either a HMM model instance or a word end. Each model node is itself a network consisting of states connected by arcs. Thus when fully compiled, a speech recognition network consists of HMM states connected by transitions. For an unknown input utterance with T frames, every path from the start node to the exit node of the network passes through T HMM states. Each of these paths has log probability which is computed by summing the log probability of each individual transition in the path and the log probability of each emitting state generating the corresponding observation. The function of the Viterbi decoder is find those paths through the network which have the highest log probability. This is found using the Token Passing algorithm. In a network that has many nodes, the computation time is reduced by only allowing propagation of those tokens which will have some chance of becoming winners. This process is called pruning.

Natural Language Processor

In a typical natural language interface to a database, the user enters a question in his/her natural language, for example, English. The system parses it and translates it to a query language expression. The system then uses the query language expression to process the query and if the search is successful, a recordset representing the results is displayed in English either formatted as raw text or in a graphical form. For a natural language interface to work well involves a number of technical requirements.

For example, it needs to be robust—in the sentence 'What's the departments turnover' it needs to decide that the word whats=what's=what is. And it also has to determine that departments=department's. In addition to being robust, the natural language interface has to distinguish between the several possible forms of ambiguity that may exist in the natural language—lexical, structural, reference and ellipsis ambiguity. All of these requirements, in addition to the general ability to perform basic linguistic morphological operations of tokenization, tagging and grouping, are implemented within the present invention.

Tokenization is implemented by a text analyzer which treats the text as a series of tokens or useful meaningful units that are larger than individual characters, but smaller than phrases and sentences. These include words, separable parts of words, and punctuation. Each token is associated with an offset and a length. The first phase of tokenization is the process of segmentation which extracts the individual tokens from the input text and keeps track of the offset where each token originated in the input text. The tokenizer output lists the offset and category for each token. In the next phase of the text analysis, the tagger uses a built-in morphological analyzer to look up each word/token in a phrase or sentence and internally lists all parts of speech. The output is the input string with each token tagged with a parts of speech notation. Finally the grouper which functions as a phrase extractor or phrase analyzer, determines which groups of words form phrases. These three operations which are the foundations for any modern linguistic processing schemes, are fully implemented in optimized algorithms for determining the single-best possible answer to the user's question.

SQL Database and Full-Text Query

Another key component of present system is a SQL-database. This database is used to store text, specifically the answer-question pairs are stored in full-text tables of the database. Additionally, the full-text search capability of the database allows full-text searches to be carried out.

While a large portion of all digitally stored information is in the form of unstructured data, primarily text, it is now possible to store this textual data in traditional database systems in character-based columns such as varchar and text. In order to effectively retrieve textual data from the database, techniques have to be implemented to issue queries against textual data and to retrieve the answers in a meaningful way where it provides the answers as in the case of the NLQS system.

There are two major types of textual searches: Property—This search technology first applies filters to documents in order to extract properties such as author, subject, type, word count, printed page count, and time last written, and then issues searches against those properties; Full-text—this search technology first creates indexes of all non-noise words in the documents, and then uses these indexes to support linguistic searches and proximity searches.

Two additional technologies are also implemented in this particular RDBMs: SQL Server also have been integrated: A Search service—a full-text indexing and search service that is called both index engine and search, and a parser that accepts full-text SQL extensions and maps them into a form that can be processed by the search engine.

The four major aspects involved in implementing full-text retrieval of plain-text data from a full-text-capable database are: Managing the definition of the tables and columns that are registered for full-text searches; Indexing the data in registered columns—the indexing process scans the character streams, determines the word boundaries (this is called word breaking), removes all noise words (this also is called stop words), and then populates a full-text index with the remaining words; Issuing queries against registered columns for populated full-text indexes; Ensuring that subsequent changes to the data in registered columns gets propagated to the index engine to keep the full-text indexes synchronized.

The underlying design principle for the indexing, querying, and synchronizing processes is the presence of a full-text unique key column (or single-column primary key) on all tables registered for full-text searches. The full-text index contains an entry for the non-noise words in each row together with the value of the key column for each row.

When processing a full-text search, the search engine returns to the database the key values of the rows that match the search criteria.

The full-text administration process starts by designating a table and its columns of interest for full-text search. Customized NLQS stored procedures are used first to register tables and columns as eligible for full-text search. After that, a separate request by means of a stored procedure is issued to populate the full-text indexes. The result is that the underlying index engine gets invoked and asynchronous index population begins. Full-text indexing tracks which significant words are used and where they are located. For example, a full-text index might indicate that the word "NLQS" is found at word number **423** and word number **982** in the Abstract column of the DevTools table for the row associated with a ProductID of 6. This index structure supports an efficient search for all items containing indexed

words as well as advanced search operations, such as phrase searches and proximity searches. (An example of a phrase search is looking for "white elephant," where "white" is followed by "elephant". An example of a proximity search is looking for "big" and "house" where "big" occurs near "house".) To prevent the full-text index from becoming bloated, noise words such as "a," "and," and "the" are ignored.

Extensions to the Transact-SQL language are used to construct full-text queries. The two key predicates that are used in the NLQS are CONTAINS and FREETEXT.

The CONTAINS predicate is used to determine whether or not values in full-text registered columns contain certain words and phrases. Specifically, this predicate is used to search for:

A word or phrase.

The prefix of a word or phrase.

A word or phrase that is near another.

A word that is an inflectional form of another (for example, "drive" is the inflectional stem of "drives," "drove," "driving," and "driven").

A set of words or phrases, each of which is assigned a different weighting.

The relational engine within SQL Server recognizes the CONTAINS and FREETEXT predicates and performs some minimal syntax and semantic checking, such as ensuring that the column referenced in the predicate has been registered for full-text searches. During query execution, a full-text predicate and other relevant information are passed to the full-text search component. After further syntax and semantic validation, the search engine is invoked and returns the set of unique key values identifying those rows in the table that satisfy the full-text search condition. In addition to the FREETEXT and CONTAINS, other predicates such as AND, LIKE, NEAR are combined to create the customized NLQS SQL construct.

Full-Text Query Architecture of the SQL Database

The full-text query architecture is comprised of the following several components—Full-Text Query component, the SQL Server Relational Engine, the Full-Text provider and the Search Engine.

The Full-Text Query component of the SQL database accept a full-text predicate or rowset-valued function from the SQL Server; transform parts of the predicate into an internal format, and sends it to Search Service, which returns the matches in a rowset. The rowset is then sent back to SQL Server. SQL Server uses this information to create the resultset that is then returned to the submitter of the query.

The SQL Server Relational Engine accepts the CONTAINS and FREETEXT predicates as well as the CONTAINSTABLE() and FREETEXTTABLE() rowset-valued functions. During parse time, this code checks for conditions such as attempting to query a column that has not been registered for full-text search. If valid, then at run time, the ft_search_condition and context information is sent to the full-text provider. Eventually, the full-text provider returns a rowset to SQL Server, which is used in any joins (specified or implied) in the original query. The Full-Text Provider parses and validates the ft_search_condition, constructs the appropriate internal representation of the full-text search condition, and then passes it to the search engine. The result is returned to the relational engine by means of a rowset of rows that satisfy ft_search_condition.

Client Side System **150**

The architecture of client-side system **150** of Natural Language Query System **100** is illustrated in greater detail in FIGS. **2**A-**2**C. Referring to FIG. **2**A, the three main processes effectuated by Client System **150** are illustrated as follows: Initialization process **200**A consisting of SRE **201**, Communication **202** and Microsoft (MS) Agent **203** routines; at FIG. **2**B an iterative process **200**B consisting of two sub-routines: a) Receive User Speech **208**—made up of SRE **204** and Communication **205**; and b) Receive Answer from Server **207**—made up of MS Speak Agent **206**, Communication **209**, Voice data file **210** and Text to Speech Engine **211**. Finally, in FIG. **2**C un-initialization process **200**C is made up of three sub-routines: SRE **212**, Communication **213**, and MS Agent **214**. Each of the above three processes are described in detail in the following paragraphs. It will be appreciated by those skilled in the art that the particular implementation for such processes and routines will vary from client platform to platform, so that in some environments such processes may be embodied in hard-coded routines executed by a dedicated DSP, while in others they may be embodied as software routines executed by a shared host processor, and in still others a combination of the two may be used.

Initialization at Client System **150**

The initialization of the Client System **150** is illustrated in FIG. **2**D and is comprised generally of 3 separate initializing processes: client-side Speech Recognition Engine **220**A, MS Agent **220**B and Communication processes **220**C.

Initialization of Speech Recognition Engine **220**A

Speech Recognition Engine **155** is initialized and configured using the routines shown in **220**A. First, an SRE COM Library is initialized. Next, memory **220** is allocated to hold Source and Coder objects, are created by a routine **221**. Loading of configuration file **221**A from configuration data file **221**B also takes place at the same time that the SRE Library is initialized. In configuration file **221**B, the type of the input of Coder and the type of the output of the Coder are declared. The structure, operation, etc. of such routines are well-known in the art, and they can be implemented using a number of fairly straightforward approaches. Accordingly, they are not discussed in detail herein. Next, Speech and Silence components of an utterance are calibrated using a routine **222**, in a procedure that is also well-known in the art. To calibrate the speech and silence components, the user preferably articulates a sentence that is displayed in a text box on the screen. The SRE library then estimates the noise and other parameters required to find silence and speech elements of future user utterances.

Initialization of MS Agent **220**B

The software code used to initialize and set up a MS Agent **220**B is also illustrated in FIG. **2**D. The MS Agent **220**B routine is responsible for coordinating and handling the actions of the animated agent **157** (FIG. **1**). This initialization thus consists of the following steps:

1. Initialize COM library **223**. This part of the code initializes the COM library, which is required to use ActiveX Controls, which controls are well-known in the art.
2. Create instance of Agent Server **224**—this part of the code creates an instance of Agent ActiveX control.
3. Loading of MS Agent **225**—this part of the code loads MS Agent character from a specified file **225**A containing general parameter data for the Agent Character, such as the overall appearance, shape, size, etc.
4. Get Character Interface **226**—this part of the code gets an appropriate interface for the specified character; for

example, characters may have different control/interaction capabilities that can be presented to the user.

5. Add Commands to Agent Character Option **227**—this part of the code adds commands to an Agent Properties sheet, which sheet can be accessed by clicking on the icon that appears in the system tray, when the Agent character is loaded e.g., that the character can Speak, how he/she moves, TTS Properties, etc.
6. Show the Agent Character **228**—this part of the code displays the Agent character on the screen so it can be seen by the user;
7. AgentNotifySink—to handle events. This part of the code creates AgentNotifySink object **229**, registers it at **230** and then gets the Agent Properties interface **231**. The property sheet for the Agent character is assigned using routine **232**.
8. Do Character Animations **233**—This part of the code plays specified character animations to welcome the user to NLQS **100**.

The above then constitutes the entire sequence required to initialize the MS Agent. As with the SRE routines, the MS Agent routines can be implemented in any suitable and conventional fashion by those skilled in the art based on the present teachings. The particular structure, operation, etc. of such routines is not critical, and thus they are not discussed in detail herein.

In a preferred embodiment, the MS Agent is configured to have an appearance and capabilities that are appropriate for the particular application. For instance, in a remote learning application, the agent has the visual form and mannerisms/attitude/gestures of a college professor. Other visual props (blackboard, textbook, etc.) may be used by the agent and presented to the user to bring to mind the experience of being in an actual educational environment. The characteristics of the agent may be configured at the client side **150**, and/or as part of code executed by a browser program (not shown) in response to configuration data and commands from a particular web page. For example, a particular website offering medical services may prefer to use a visual image of a doctor. These and many other variations will be apparent to those skilled in the art for enhancing the human-like, real-time dialog experience for users.

Initialization of Communication Link **160**A

The initialization of Communication Link **160**A is shown with reference to process **220**C FIG. **2**D. Referring to FIG. **2**D, this initialization consists of the following code components: Open INTERNET Connection **234**—this part of the code opens an INTERNET Connection and sets the parameter for the connection. Then Set Callback Status routine **235** sets the callback status so as to inform the user of the status of connection. Finally Start New HTTP INTERNET Session **236** starts a new INTERNET session. The details of Communications Link **160** and the set up process **220**C are not critical, and will vary from platform to platform. Again, in some cases, users may use a low-speed dial-up connection, a dedicated high speed switched connection (T1 for example), an always-on xDSL connection, a wireless connection, and the like.

Iterative Processing of Queries/Answers

As illustrated in FIG. **3**, once initialization is complete, an iterative query/answer process is launched when the user presses the Start Button to initiate a query. Referring to FIG. **3**, the iterative query/answer process consists of two main sub-processes implemented as routines on the client side system **150**: Receive User Speech **240** and Receive User Answer **243**. The Receive User Speech **240** routine receives speech from the user (or another audio input source), while the Receive User Answer **243** routine receives an answer to the user's question in the form of text from the server so that it can be converted to speech for the user by text-to-speech engine **159**. As used herein, the term "query" is referred to in the broadest sense to refer, to either a question, a command, or some form of input used as a control variable by the system. For example, a query may consist of a question directed to a particular topic, such as "what is a network" in the context of a remote learning application. In an e-commerce application a query might consist of a command to "list all books by Mark Twain" for example. Similarly, while the answer in a remote learning application consists of text that is rendered into audible form by the text to speech engine **159**, it could also be returned as another form of multi-media information, such as a graphic image, a sound file, a video file, etc. depending on the requirements of the particular application. Again, given the present teachings concerning the necessary structure, operation, functions, performance, etc., of the client-side Receive User Speech **240** and Receiver User Answer **243** routines, one of ordinary skill in the art could implement such in a variety of ways.

Receive User Speech—As illustrated in FIG. **3**, the Receive User Speech routine **240** consists of a SRE **241** and a Communication **242** process, both implemented again as routines on the client side system **150** for receiving and partially processing the user's utterance. SRE routine **241** uses a coder **248** which is prepared so that a coder object receives speech data from a source object. Next the Start Source **249** routine is initiated. This part of the code initiates data retrieval using the source Object which will in turn be given to the Coder object. Next, MFCC vectors **250** are extracted from the Speech utterance continuously until silence is detected. As alluded to earlier, this represents the first phase of processing of the input speech signal, and in a preferred embodiment, it is intentionally restricted to merely computing the MFCC vectors for the reasons already expressed above. These vectors include the 12 cepstral coefficients and the RMS energy term, for a total of 13 separate numerical values for the partially processed speech signal.

In some environments, nonetheless, it is conceivable that the MFCC delta parameters and MFCC acceleration parameters can also be computed at client side system **150**, depending on the computation resources available, the transmission bandwidth in data link **160**A available to server side system **180**, the speed of a transceiver used for carrying data in the data link, etc. These parameters can be determined automatically by client side system upon initializing SRE **155** (using some type of calibration routine to measure resources), or by direct user control, so that the partitioning of signal processing responsibilities can be optimized on a case-by-case basis. In some applications, too, server side system **180** may lack the appropriate resources or routines for completing the processing of the speech input signal. Therefore, for some applications, the allocation of signal processing responsibilities may be partitioned differently, to the point where in fact both phases of the speech signal processing may take place at client side system **150** so that the speech signal is completely—rather than partially—processed and transmitted for conversion into a query at server side system **180**.

Again in a preferred embodiment, to ensure reasonable accuracy and real-time performance from a query/response perspective, sufficient resources are made available in a client side system so that 100 frames per second of speech

data can be partially processed and transmitted through link **160**A. Since the least amount of information that is necessary to complete the speech recognition process (only 13 coefficients) is sent, the system achieves a real-time performance that is believed to be highly optimized, because other latencies (i.e., client-side computational latencies, packet formation latencies, transmission latencies) are minimized. It will be apparent that the principles of the present invention can be extended to other SR applications where some other methodology is used for breaking down the speech input signal by an SRE (i.e., non-MFCC based). The only criteria is that the SR processing be similarly dividable into multiple phases, and with the responsibility for different phases being handled on opposite sides of link **160**A depending on overall system performance goals, requirements and the like. This functionality of the present invention can thus be achieved on a system-by-system basis, with an expected and typical amount of optimization being necessary for each particular implementation.

Thus, the present invention achieves a response rate performance that is tailored in accordance with the amount of information that is computed, coded and transmitted by the client side system **150**. So in applications where real-time performance is most critical, the least possible amount of extracted speech data is transmitted to reduce these latencies, and, in other applications, the amount of extracted speech data that is processed, coded and transmitted can be varied.

Communication—transmit communication module **242** is used to implement the transport of data from the client to the server over the data link **160**A, which in a preferred embodiment is the INTERNET. As explained above, the data consists of encoded MFCC vectors that will be used at then server-side of the Speech Recognition engine to complete the speech recognition decoding. The sequence of the communication is as follows:

OpenHTTPRequest **251**—this part of the code first converts MFCC vectors to a stream of bytes, and then processes the bytes so that it is compatible with a protocol known as HTTP. This protocol is well-known in the art, and it is apparent that for other data links another suitable protocol would be used.

1. Encode MFCC Byte Stream **251**—this part of the code encodes the MFCC vectors, so that they can be sent to the server via HTTP.
2. Send data **252**—this part of the code sends MFCC vectors to the server using the INTERNET connection and the HTTP protocol.

Wait for the Server Response **253**—this part of the code monitors the data link **160**A a response from server side system **180** arrives. In summary, the MFCC parameters are extracted or observed on-the-fly from the input speech signal. They are then encoded to a HTTP byte stream and sent in a streaming fashion to the server before the silence is detected—i.e. sent to server side system **180** before the utterance is complete. This aspect of the invention also facilitates a real-time behavior, since data can be transmitted and processed even while the user is still speaking.

Receive Answer from Server **243** is comprised of the following modules as shown in FIG. **3**.: MS Agent **244**, Text-to-Speech Engine **245** and receive communication modules **246**. All three modules interact to receive the answer from server side system **180**. As illustrated in FIG. **3**, the receive communication process consists of three separate processes implemented as a receive routine on client side system **150**: a Receive the Best Answer **258** receives the best answer over data link **160**B (the HTTP communication channel). The answer is de-compressed at **259** and then the answer is passed by code **260** to the MS Agent **244**, where it is received by code portion **254**. A routine **255** then articulates the answer using text-to-speech engine **257**. Of course, the text can also be displayed for additional feedback purposes on a monitor used with client side system **150**. The text to speech engine uses a natural language voice data As file **256** associated with it that is appropriate for the particular language application (i.e., English, French, German, Japanese, etc.). As explained earlier when the answer is something more than text, it can be treated as desired to provide responsive information to the user, such as with a graphics image, a sound, a video clip, etc.

Uninitialization

The un-initialization routines and processes are illustrated in FIG. **4**. Three functional modules are used for un-initializing the primary components of the client side system **150**; these include SRE **270**, Communications **271** and MS Agent **272** un-initializing routines. To un-initialize SRE **220**A, memory that was allocated in the initialization phase is de-allocated by code **273** and objects created during such initialization phase are deleted by code **274**. Similarly, as illustrated in FIG. **4**, to un-initialize Communications module **220**C the INTERNET connection previously established with the server is closed by code portion **275** of the Communication Un-initialization routine **271**. Next the INTERNET session created at the time of initialization is also closed by routine **276**. For the un-initialization of the MS Agent **220**B, as illustrated in FIG. **4**, MS Agent Un-initialization routine **272** first releases the Commands Interface **227** using routine **277**. This releases the commands added to the property sheet during loading of the agent character by routine **225**. Next the Character Interface initialized by routine **226** is released by routine **278** and the Agent is unloaded at **279**. The Sink Object Interface is then also released **280** followed by the release of the Property Sheet Interface **281**. The Agent Notify Sink **282** then un-registers the Agent and finally the Agent Interface **283** is released which releases all the resources allocated during initialization steps identified in FIG. **2**D.

It will be appreciated by those skilled in the art that the particular implementation for such un-initialization processes and routines in FIG. **4** will vary from client platform to client platform, as for the other routines discussed above. The structure, operation, etc. of such routines are well-known in the art, and they can be implemented using a number of fairly straightforward approaches without undue effort. Accordingly, they are not discussed in detail herein.

Description of Server Side System **180**

Introduction

A high level flow diagram of the set of preferred processes implemented on server side system **180** of Natural Language Query System **100** is illustrated in FIG. **11**A through FIG. **11**C. In a preferred embodiment, this process consists of a two step algorithm for completing the processing of the speech input signal, recognizing the meaning of the user's query, and retrieving an appropriate answer/response for such query.

The 1st step as illustrated in FIG. **11**A can be considered a high-speed first-cut pruning mechanism, and includes the following operations: after completing processing of the speech input signal, the user's query is recognized at step **1101**, so that the text of the query is simultaneously sent to

Natural Language Engine **190** (FIG. **1**) at step **1107**, and to DB Engine **186** (also FIG. **1**) at step **1102**. By "recognized" in this context it is meant that the user's query is converted into a text string of distinct native language words through the HMM technique discussed earlier.

At NLE **190**, the text string undergoes morphological linguistic processing at step **1108**: the string is tokenized the tags are tagged and the tagged tokens are grouped Next the noun phrases (NP) of the string are stored at **1109**, and also copied and transferred for use by DB Engine **186** during a DB Process at step **1110**. As illustrated in FIG. **11**A, the string corresponding to the user's query which was sent to the DB Engine **186** at **1102**, is used together with the NP received from NLE **190** to construct an SQL Query at step **1103**. Next, the SQL query is executed at step **1104**, and a record set of potential questions corresponding to the user's query are received as a result of a full-text search at **1105**, which are then sent back to NLE **190** in the form of an array at step **1106**.

As can be seen from the above, this first step on the server side processing acts as an efficient and fast pruning mechanism so that the universe of potential "hits" corresponding to the user's actual query is narrowed down very quickly to a manageable set of likely candidates in a very short period of time.

Referring to FIG. **11**B, in contrast to the first step above, the 2^{nd} step can be considered as the more precise selection portion of the recognition process. It begins with linguistic processing of each of the stored questions in the array returned by the full-text search process as possible candidates representing the user's query. Processing of these stored questions continues in NLE **190** as follows: each question in the array of questions corresponding to the record set returned by the SQL full-text search undergoes morphological linguistic processing at step **1111**: in this operation, a text string corresponding to the retrieved candidate question is tokenized, the tags are tagged and the tagged tokens are grouped. Next, noun phrases of the string are computed and stored at step **1112**. This process continues iteratively at point **1113**, and the sequence of steps at **1118**, **1111**, **1112**, **1113** are repeated so that an NP for each retrieved candidate question is computed and stored. Once an NP is computed for each of the retrieved candidate questions of the array, a comparison is made between each such retrieved candidate question and the user's query based on the magnitude of the NP value at step **1114**. This process is also iterative in that steps **1114**, **1115**, **1116**, **1119** are repeated so that the comparison of the NP for each retrieved candidate question with that of the NP of the user's query is completed. When there are no more stored questions in the array to be processed at step **1117**, the stored question that has the maximum NP relative to the user's query, is identified at **1117**A as the stored question which best matches the user's query.

Notably, it can be seen that the second step of the recognition process is much more computationally intensive than the first step above, because several text strings are tokenized, and a comparison is made of several NPs. This would not be practical, nonetheless, if it were not for the fact that the first step has already quickly and efficiently reduced the candidates to be evaluated to a significant degree. Thus, this more computationally intensive aspect of the present invention is extremely valuable, however because it yields extremely high accuracy in the overall query recognition process. In this regard, therefore, this second step of the query recognition helps to ensure the overall accuracy of the system, while the first step helps to maintain a satisfactory speed that provides a real-time feel for the user.

As illustrated in FIG. **11**C, the last part of the query/response process occurs by providing an appropriate matching answer/response to the user. Thus, an identity of a matching stored question is completed at step **1120**. Next a file path corresponding to an answer of the identified matching question is extracted at step **1121**. Processing continues so that the answer is extracted from the file path at **1122** and finally the answer is compressed and sent to client side system **150** at step **1123**.

The discussion above is intended to convey a general overview of the primary components, operations, functions and characteristics of those portions of NLQS system **100** that reside on server side system **180**. The discussion that follows describes in more detail the respective sub-systems.

Software Modules used in Server Side System **180**

The key software modules used on server-side system **180** of the NLQS system are illustrated in FIG. **5**. These include generally the following components: a Communication module **500**—identified as CommunicationServer ISAPI **500**A (which is executed by SRE Server-side **182**—FIG. **1** and is explained in more detail below), and a database process DBProcess module **501** (executed by DB Engine **186**—FIG. **1**). Natural language engine module **500**C (executed by NLE **190**—FIG. **1**) and an interface **500**B between the NLE process module **500**C and the DBProcess module **500**B. As shown here, CommunicationServerISAPI **500**A includes a server-side speech recognition engine and appropriate communication interfaces required between client side system **150** and server side system **180**. As further illustrated in FIG. **5**, server-side logic of Natural Language Query System **100** also can be characterized as including two dynamic link library components: CommunicationServerISAPI **500** and DBProcess **501**. The CommunicationServerIASPI **500** is comprised of 3 sub-modules: Server-side Speech Recognition Engine module **500**A; Interface module **500**B between Natural Language Engine modules **500**C and DBProcess **501**; and the Natural Language Engine modules **500**C.

DB Process **501** is a module whose primary function is to connect to a SQL database and to execute an SQL query that is composed in response to the user's query. In addition, this module interfaces with logic that fetches the correct answer from a file path once this answer is passed to it from the Natural Language Engine module **500**C.

Speech Recognition Sub-System **182** on Server-Side System **180**

The server side speech recognition engine module **500**A is a set of distributed components that perform the necessary functions and operations of speech recognition engine **182** (FIG. **1**) at server-side **180**. These components can be implemented as software routines that are executed by server side **180** in conventional fashion. Referring to FIG. **4**A, a more detailed break out of the operation of the speech recognition components **600** at the server-side can be seen as follows:

Within a portion **601** of the server side SRE module **500**A, the binary MFCC vector byte stream corresponding to the speech signal's acoustic features extracted at client side system **150** and sent over the communication channel **160** is received. The MFCC acoustic vectors are decoded from the encoded HTTP byte stream as follows: Since the MFCC vectors contain embedded NULL characters, they cannot be transferred in this form to server side system **180** as such using HTTP protocol. Thus the MFCC vectors are first

encoded at client-side **150** before transmission in such a way that all the speech data is converted into a stream of bytes without embedded NULL characters in the data. At the very end of the byte stream a single NULL character is introduced to indicate the termination of the stream of bytes to be transferred to the server over the INTERNET **160**A using HTTP protocol.

As explained earlier, to conserve latency time between the client and server, a smaller number of bytes (just the 13 MFCC coefficients) are sent from client side system **150** to server side system **180**. This is done automatically for each platform to ensure uniformity, or can be tailored by the particular application environment—i.e., such as where it is determined that it will take less time to compute the delta and acceleration coefficients at the server (26 more calculations), than it would take to encode them at the client, transmit them, and then decode them from the HTTP stream. In general, since server side system **180** is usually better equipped to calculate the MFCC delta and acceleration parameters, this is a preferable choice. Furthermore, there is generally more control over server resources compared to the client's resources, which means that future upgrades, optimizations, etc., can be disseminated and shared by all to make overall system performance more reliable and predictable. So, the present invention can accommodate even the worst-case scenario where the client's machine may be quite thin and may just have enough resources to capture the speech input data and do minimal processing.

Dictionary Preparation & Grammar Files

Referring to FIG. **4**A, within code block **605**, various options selected by the user (or gleaned from the user's status within a particular application) are received. For instance, in the case of a preferred remote learning system, Course, Chapter and/or Section data items are communicated. In the case of other applications (such as e-commerce) other data options are communicated, such as the Product Class, Product Category, Product Brand, etc. loaded for viewing within his/her browser. These selected options are based on the context experienced by the user during an interactive process, and thus help to limit and define the scope—i.e. grammars and dictionaries that will be dynamically loaded to speech recognition engine **182** (FIG. **1**) for Viterbi decoding during processing of the user speech utterance. For speech recognition to be optimized both grammar and dictionary files are used in a preferred embodiment. A Grammar file supplies the universe of available user queries; i.e., all the possible words that are to be recognized. The Dictionary file provides phonemes (the information of how a word is pronounced—this depends on the specific native language files that are installed—for example, UK English or US English) of each word contained in the grammar file. It is apparent that if all the sentences for a given environment that can be recognized were contained in a single grammar file then recognition accuracy would be deteriorated and the loading time alone for such grammar and dictionary files would impair the speed of the speech recognition process.

To avoid these problems, specific grammars are dynamically loaded or actively configured as the current grammar according to the user's context, i.e., as in the case of a remote learning system, the Course, Chapter and/or Section selected. Thus the grammar and dictionary files are loaded dynamically according to the given Course, Chapter and/or Section as dictated by the user, or as determined automatically by an application program executed by the user.

The second code block **602** implements the initialization of Speech Recognition engine **182** (FIG. **1**). The MFCC vectors received from client side system **150** along with the grammar filename and the dictionary file names are introduced to this block to initialize the speech decoder.

As illustrated in FIG. **4**A, the initialization process **602** uses the following sub-routines: A routine **602***a* for loading an SRE library. This then allows the creation of an object identified as External Source with code **602***b* using the received MFCC vectors. Code **602***c* allocates memory to hold the recognition objects. Routine **602***d* then also creates and initializes objects that are required for the recognition such as: Source, Coder, Recognizer and Results Loading of the Dictionary created by code **602***e*, Hidden Markov Models (HMMs) generated with code **602***f*; and Loading of the Grammar file generated by routine **602***g*.

Speech Recognition **603** is the next routine invoked as illustrated in FIG. **4**A, and is generally responsible for completing the processing of the user speech signals input on the client side **150**, which, as mentioned above, are preferably only partially processed (i.e., only MFCC vectors are computed during the first phase) when they are transmitted across link **160**. Using the functions created in External Source by subroutine **602***b*, this code reads MFCC vectors, one at a time from an External Source **603***a*, and processes them in block **603***b* to realize the words in the speech pattern that are symbolized by the MFCC vectors captured at the client. During this second phase, an additional 13 delta coefficients and an additional 13 acceleration coefficients are computed as part of the recognition process to obtain a total of 39 observation vectors O_t referred to earlier. Then, using a set of previously defined Hidden Markov Models (HMMs), the words corresponding to the user's speech utterance are determined in the manner described earlier. This completes the word "recognition" aspect of the query processing, which results are used further below to complete the query processing operations.

It will be appreciated by those skilled in the art that the distributed nature and rapid performance of the word recognition process, by itself, is extremely useful and may be implemented in connection with other environments that do not implicate or require additional query processing operations. For example, some applications may simply use individual recognized words for filling in data items on a computer generated form, and the aforementioned systems and processes can provide a rapid, reliable mechanism for doing so.

Once the user's speech is recognized, the flow of SRE 182 passes to Un-initialize SRE routine **604** where the speech engine is un-initialized as illustrated. In this block all the objects created in the initialization block are deleted by routine **604***a*, and memory allocated in the initialization block during the initialization phase are removed by routine **604***b*.

Again, it should be emphasized that the above are merely illustrative of embodiments for implementing the particular routines used on a server side speech recognition system of the present invention. Other variations of the same that achieve the desired functionality and objectives of the present invention will be apparent from the present teachings.

Database Processor **186** Operation—DBProcess

Construction of an SQL Query used as part of the user query processing is illustrated in FIG. **4**B, a SELECT SQL statement is preferably constructed using a conventional CONTAINS predicate. Module **950** constructs the SQL query based on this SELECT SQL statement, which query is used for retrieving the best suitable question stored in the

database corresponding to the user's articulated query, (designated as Question here). A routine **951** then concatenates a table name with the constructed SELECT statement. Next, the number of words present in each Noun Phrase of Question asked by the user is calculated by routine **952**. Then memory is allocated by routine **953** as needed to accommodate all the words present in the NP. Next a word List (identifying all the distinct words present in the NP) is obtained by routine **954**. After this, this set of distinct words are concatenated by routine **955** to the SQL Query separated with a NEAR ( ) keyword. Next, the AND keyword is concatenated to the SQL Query by routine **956** after each NP. Finally memory resources are freed by code **957** so as to allocate memory to store the words received from NP for any next iteration. Thus, at the end of this process, a completed SQL Query corresponding to the user's articulated question is generated.

Connection to SQL Server—As illustrated in FIG. **4**C, after the SQL Query is constructed by routine **710**, a routine **711** implements a connection to the query database **717** to continue processing of the user query. The connection sequence and the subsequent retrieved record set is implemented using routines **700** which include the following:

1. Server and database names are assigned by routine **711**A to a DBProcess member variable
2. A connection string is established by routine **711**B;
3. The SQL Server database is connected under control of code **711**C
4. The SQL Query is received by routine **712**A
5. The SQL Query is executed by code **712**B
6. Extract the total number of records retrieved by the query—**713**
7. Allocate the memory to store the total number of paired questions—**713**
8. Store the entire number of paired questions into an array—**713**

Once the Best Answer ID is received at **716** FIG. **4**C, from the NLE **14** (FIG. **5**), the code corresponding **716**C receives it passes it to code in **716**B where the path of the Answer file is determined using the record number. Then the file is opened **716**C using the path passed to it and the contents of the file corresponding to the answer is read. Then the answer is compressed by code in **716**D and prepared for transmission over the communication channel **160**B (FIG. **1**).

NLQS Database **188**—Table Organization

FIG. **6** illustrates a preferred embodiment of a logical structure of tables used in a typical NLQS database **188** (FIG. **1**). When NLQS database **188** is used as part of NLQS query system **100** implemented as a remote learning/training environment, this database will include an organizational multi-level hierarchy that consists typically of a Course **701**, which is made of several chapters **702**, **703**, **704**. Each of these chapters can have one or more Sections **705**, **706**, **707** as shown for Chapter 1. A similar structure can exist for Chapter 2, Chapter 3 . . . Chapter N. Each section has a set of one or more question—answer pairs **708** stored in tables described in more detail below. While this is an appropriate and preferable arrangement for a training/learning application, it is apparent that other implementations would be possible and perhaps more suitable for other applications such as e-commerce, e-support, INTERNET browsing, etc., depending on overall system parameters.

It can be seen that the NLQS database **188** organization is intricately linked to the switched grammar architecture described earlier. In other words, the context (or environment) experienced by the user can be determined at any moment in time based at the selection made at the section level, so that only a limited subset of question-answer pairs **708** for example are appropriate for section **705**. This in turn means that only a particular appropriate grammar for such question-answer pairs may be switched in for handling user queries while the user is experiencing such context. In a similar fashion, an e-commerce application for an INTERNET based business may consist of a hierarchy that includes a first level "home" page **701** identifying user selectable options (product types, services, contact information, etc.), a second level may include one or more "product types" pages **702**, **703**, **704**, a third page may include particular product models **705**, **706**, **707**, etc., and with appropriate question-answer pairs **708** and grammars customized for handling queries for such product models. Again, the particular implementation will vary from application to application, depending on the needs and desires of such business, and a typical amount of routine optimization will be necessary for each such application.

Table Organization

In a preferred embodiment, an independent database is used for each Course. Each database in turn can include three types of tables as follows: a Master Table as illustrated in FIG. **7**A, at least one Chapter Table as illustrated in FIG. **7**B and at least one Section Table as illustrated in FIG. **7**C.

As illustrated in FIG. **7**A, a preferred embodiment of a Master Table has six columns—Field Name **701**A, Data Type **702**A, Size **703**A, Null **704**A, Primary Key **705**A and Indexed **706**A. These parameters are well-known in the art of database design and structure. The Master Table has only two fields—Chapter Name **707**A and Section Name **708**A. Both ChapterName and Section Name are commonly indexed.

A preferred embodiment of a Chapter Table is illustrated in FIG. **7**B. As with the Master Table, the Chapter Table has six (6) columns—Field Name **720**, Data Type **721**, Size **722**, Null **723**, Primary Key **724** and Indexed **725**. There are nine (9) rows of data however, in this case,—Chapter_ID **726**, Answer_ID **727**, Section Name **728**, Answer_Title **729**, PairedQuestion **730**, AnswerPath **731**, Creator **732**, Date of Creation **733** and Date of Modification **734**.

An explanation of the Chapter Table fields is provided in FIG. **7**C. Each of the eight (8) Fields **720** has a description **735** and stores data corresponding to:

- AnswerID **727**—an integer that is automatically incremented for each answer given for user convenience
- Section_Name **728**—the name of the section to which the particular record belongs. This field along with the AnswerID is used as the primary key
- Answer_Title **729**—A short description of the title of the answer to the user query
- PairedQuestion **730**—Contains one or more combinations of questions for the related answers whose path is stored in the next column AnswerPath
- AnswerPath **731**—contains the path of a file, which contains the answer to the related questions stored in the previous column; in the case of a pure question/answer application, this file is a text file, but, as mentioned above, could be a multi-media file of any kind transportable over the data link **160**
- Creator **732**—Name of Content Creator
- Date_of_Creation **733**—Date on which content was created
- Date of Modification **734**—Date on which content was changed or modified

A preferred embodiment of a Section Table is illustrated in FIG. **7**D. The Section Table has six (6) columns—Field Name **740**, Data Type **741**, Size **742**, Null **743**, Primary Key **744** and Indexed **745**. There are seven (7) rows of data—Answer_ID **746**, Answer_Title **747**, PairedQuestion **748**, AnswerPath **749**, Creator **750**, Date of Creation **751** and Date of Modification **752**. These names correspond to the same fields, columns already described above for the Master Table and Chapter Table.

Again, this is a preferred approach for the specific type of learning/training application described herein. Since the number of potential applications for the present invention is quite large, and each application can be customized, it is expected that other applications (including other learning/training applications) will require and/or be better accommodated by another table, column, and field structure/hierarchy.

Search Service and Search Engine—A query text search service is performed by an SQL Search System **1000** shown in FIG. **10**. This system provides querying support to process full-text searches. This is where full-text indexes reside.

In general, SQL Search System determines which entries in a database index meet selection criteria specified by a particular text query that is constructed in accordance with an articulated user speech utterance. The Index Engine **1011**B is the entity that populates the Full-Text Index tables with indexes which correspond to the indexable units of text for the stored questions and corresponding answers. It scans through character strings, determines word boundaries, removes all noise words and then populates the full-text index with the remaining words. For each entry in the full text database that meets the selection criteria, a unique key column value and a ranking value are returned as well. Catalog set **1013** is a file-system directory that is accessible only by an Administrator and Search Service **1010**. Full-text indexes **1014** are organized into full-text catalogs, which are referenced by easy to handle names. Typically, full-text index data for an entire database is placed into a single full-text catalog.

The schema for the full-text database as described (FIG. **7**, FIG. **7**A, FIG. **7**B, FIG. **7**C, FIG. **7**D) is stored in the tables **1006** shown in FIG. **10**. Take for example, the tables required to describe the structure the stored question/answer pairs required for a particular course. For each table—Course Table, Chapter Table, Section Table, there are fields—column information that define each parameters that make up the logical structure of the table. This information is stored in User and System tables **1006**. The key values corresponding to those tables are stored as Full-Text catalogs **1013**. So when processing a full-text search, the search engine returns to the SQL Server the key values of the rows that match the search criteria. The relational engine then uses this information to respond to the query.

As illustrated in FIG. **10**, a Full-Text Query Process is implemented as follows:

1. A query **1001** that uses a SQL full-text construct generated by DB processor **186** is submitted to SQL Relational Engine **1002**.
2. Queries containing either a CONTAINS or FREETEXT predicate are rewritten by routine **1003** so that a responsive rowset returned later from Full-Text Provider **1007** will be automatically joined to the table that the predicate is acting upon. This rewrite is a mechanism used to ensure that these predicates are a seamless extension to a traditional SQL Server. After the compiled query is internally rewritten and checked for correctness in item **1003**, the query is passed to RUN TIME module **1004**. The function of module **1004** is to convert the rewritten SQL construct to a validated run-time process before it is sent to the Full-Text Provider, **1007**.
3. After this, Full-Text Provider **1007** is invoked, passing the following information for the query:
   a. A ft_search_condition parameter (this is a logical flag indicating a full text search condition)
   b. A name of a full-text catalog where a full-text index of a table resides
   c. A locale ID to be used for language (for example, word breaking)
   d. Identities of a database, table, and column to be used in the query
   e. If the query is comprised of more than one full-text construct; when this is the case Full-text provider **1007** is invoked separately for each construct.
4. SQL Relational Engine **1002** does not examine the contents of ft_search_condition. Instead, this information is passed along to Full-text provider **1007**, which verifies the validity of the query and then creates an appropriate internal representation of the full-text search condition.
5. The query request/command **1008** is then passed to Querying Support **1011**A.
6. Querying Support **1012** returns a rowset **1009** from Full-Text Catalog **1013** that contains unique key column values for any rows that match the full-text search criteria. A rank value also is returned for each row.
7. The rowset of key column values **1009** is passed to SQL Relational Engine **1002**. If processing of the query implicates either a CONTAINSTABLE() or FREETEXTTABLE() function, RANK values are returned; otherwise, any rank value is filtered out.
8. The rowset values **1009** are plugged into the initial query with values obtained from relational database **1006**, and a result set **1015** is then returned for further processing to yield a response to the user.

At this stage of the query recognition process, the speech utterance by the user has already been rapidly converted into a carefully crafted text query, and this text query has been initially processed so that an initial matching set of results can be further evaluated for a final determination of the appropriate matching question/answer pair. The underlying principle that makes this possible is the presence of a full-text unique key column for each table that is registered for full-text searches. Thus when processing a full-text search, SQL Search Service **1010** returns to SQL server **1002** the key values of the rows that match the database. In maintaining these full-text databases **1013** and full text indexes **1014**, the present invention has the unique characteristic that the full-text indices **1014** are not updated instantly when the full-text registered columns are updated. This operation is eliminated, again, to reduce recognition latency, increase response speed, etc. Thus, as compared to other database architectures, this updating of the full-text index tables, which would otherwise take a significant time, is instead done asynchronously at a more convenient time.

Interface between NLE **190** and DB Processor **188**

The result set **1015** of candidate questions corresponding to the user query utterance are presented to NLE **190** for further processing as shown in FIG. **4**D to determine a "best" matching question/answer pair. An NLE/DBProcessor interface module coordinates the handling of user queries, analysis of noun-phrases (NPs) of retrieved questions sets from the SQL query based on the user query, comparing the retrieved question NPs with the user query NP, etc.

between NLE **190** and DB Processor **188**. So, this part of the server side code contains functions, which interface processes resident in both NLE block **190** and DB Processor block **188**. The functions are illustrated in FIG. **4**D; As seen here, code routine **880** implements functions to extract the Noun Phrase (NP) list from the user's question. This part of the code interacts with NLE **190** and gets the list of Noun Phrases in a sentence articulated by the user. Similarly, Routine **813** retrieves an NP list from the list of corresponding candidate/paired questions **1015** and stores these questions into an (ranked by NP value) array. Thus, at this point, NP data has been generated for the user query, as well as for the candidate questions **1015**. As an example of determining the noun phrases of a sentence such as: "What issues have guided the President in considering the impact of foreign trade policy on American businesses?" NLE **190** would return the following as noun phrases: President, issues, impact of foreign trade policy, American businesses, impact, impact of foreign trade, foreign trade, foreign trade policy, trade, trade policy, policy, businesses. The methodology used by NLE **190** will thus be apparent to those skilled in the art from this set of noun phrases and noun sub-phrases generated in response to the example query.

Next, a function identified as Get Best Answer ID **815** is implemented. This part of the code gets a best answer ID corresponding to the user's query. To do this, routines **813**A, **813**B first find out the number of Noun phrases for each entry in the retrieved set **1015** that match with the Noun phrases in the user's query. Then routine **815***a* selects a final result record from the candidate retrieved set **1015** that contains the maximum number of matching Noun phrases.

Conventionally, nouns are commonly thought of as "naming" words, and specifically as the names of "people, places, or things". Nouns such as John, London, and computer certainly fit this description, but the types of words classified by the present invention as nouns is much broader than this. Nouns can also denote abstract and intangible concepts such as birth, happiness, evolution, technology, management, imagination, revenge, politics, hope, cookery, sport, and literacy. Because of the enormous diversity of nouns compared to other parts of speech, the Applicant has found that it is much more relevant to consider the noun phrase as a key linguistic metric. So, the great variety of items classified as nouns by the present invention helps to discriminate and identify individual speech utterances much easier and faster than prior techniques disclosed in the art.

Following this same thought, the present invention also adopts and implements another linguistic entity—the word phrase—to facilitate speech query recognition. The basic structure of a word phrase—whether it be a noun phrase, verb phrase, adjective phrase—is three parts—[pre-Head string],[Head] and [post-Head string]. For example, in the minimal noun phrase—"the children," "children" is classified as the Head of the noun phrase. In summary, because of the diversity and frequency of noun phrases, the choice of noun phrase as the metric by which stored answer is linguistically chosen, has a solid justification in applying this technique to the English natural language as well as other natural languages. So, in sum, the total noun phrases in a speech utterance taken together operate extremely well as unique type of speech query fingerprint.

The ID corresponding to the best answer corresponding to the selected final result record question is then generated by routine **815** which then returns it to DB Process shown in FIG. **4**C. As seen there, a Best Answer ID I is received by routine **716**A, and used by a routine **716**B to retrieve an answer file path. Routine **716**C then opens and reads the answer file, and communicates the substance of the same to routine **716**D. The latter then compresses the answer file data, and sends it over data link **160** to client side system **150** for processing as noted earlier (i.e., to be rendered into audible feedback, visual text/graphics, etc.). Again, in the context of a learning/instructional application, the answer file may consist solely of a single text phrase, but in other applications the substance and format will be tailored to a specific question in an appropriate fashion. For instance, an "answer" may consist of a list of multiple entries corresponding to a list of responsive category items (i.e., a list of books to a particular author) etc. Other variations will be apparent depending on the particular environment.

Natural Language Engine **190**

Again referring to FIG. **4**D, the general structure of NL engine **190** is depicted. This engine implements the word analysis or morphological analysis of words that make up the user's query, as well as phrase analysis of phrases extracted from the query.

As illustrated in FIG. **9**, the functions used in a morphological analysis include tokenizers **802**A, stemmers **804**A and morphological analyzers **806**A. The functions that comprise the phrase analysis include tokenizers, taggers and groupers, and their relationship is shown in FIG. **8**.

Tokenizer **802**A is a software module that functions to break up text of an input sentence **801**A into a list of tokens **803**A. In performing this function, tokenizer **802**A goes through input text **801**A and treats it as a series of tokens or useful meaningful units that are typically larger than individual characters, but smaller than phrases and sentences. These tokens **803**A can include words, separable parts of word and punctuation. Each token **803**A is given an offset and a length. The first phase of tokenization is segmentation, which extracts the individual tokens from the input text and keeps track of the offset where each token originated from in the input text. Next, categories are associated with each token, based on its shape. The process of tokenization is well-known in the art, so it can be performed by any convenient application suitable for the present invention.

Following tokenization, a stemmer process **804**A is executed, which can include two separate forms—inflectional and derivational, for analyzing the tokens to determine their respective stems **805**A. An inflectional stemmer recognizes affixes and returns the word which is the stem. A derivational stemmer on the other hand recognizes derivational affixes and returns the root word or words. While stemmer **804**A associates an input word with its stem, it does not have parts of speech information. Analyzer **806**B takes a word independent of context, and returns a set of possible parts of speech **806**A.

As illustrated in FIG. **8**, phrase analysis **800** is the next step that is performed after tokenization. A tokenizer **802** generates tokens from input text **801**. Tokens **803** are assigned to parts of a speech tag by a tagger routine **804**, and a grouper routine **806** recognizes groups of words as phrases of a certain syntactic type. These syntactic types include for example the noun phrases mentioned earlier, but could include other types if desired such as verb phrases and adjective phrases. Specifically, tagger **804** is a parts-of-speech disambiguator, which analyzes words in context. It has a built-in morphological analyzer (not shown) that allows it to identify all possible parts of speech for each token. The output of tagger **804** is a string with each token tagged with a parts-of-speech label **805**. The final step in the linguistic process **800** is the grouping of words to form phrases **807**. This function is performed by the grouper **806**,

and is very dependent, of course, on the performance and output of tagger component **804**.

Accordingly, at the end of linguistic processing **800**, a list of noun phrases (NP) **807** is generated in accordance with the user's query utterance. This set of NPs generated by NLE **190** helps significantly to refine the search for the best answer, so that a single-best answer can be later provided for the user's question.

The particular components of NLE **190** are shown in FIG. **4**D, and include several components. Each of these components implement the several different functions required in NLE **190** as now explained.

Initialize Grouper Resources Object and the Library **900**—this routine initializes the structure variables required to create grouper resource object and library. Specifically, it initializes a particular natural language used by NLE **190** to create a Noun Phrase, for example the English natural language is initialized for a system that serves the English language market. In turn, it also creates the objects (routines) required for Tokenizer, Tagger and Grouper (discussed above) with routines **900**A, **900**B, **900**C and **900**D respectively, and initializes these objects with appropriate values. It also allocates memory to store all the recognized Noun Phrases for the retrieved question pairs.

Tokenizing of the words from the given text (from the query or the paired questions) is performed with routine **909**B—here all the words are tokenized with the help of a local dictionary used by NLE **190** resources. The resultant tokenized words are passed to a Tagger routine **909**C. At routine **909**C, tagging of all the tokens is done and the output is passed to a Grouper routine **909**D.

The Grouping of all tagged token to form NP list is implemented by routine **909**D so that the Grouper groups all the tagged token words and outputs the Noun Phrases.

Un-initializing of the grouper resources object and freeing of the resources, is performed by routines **909**EA, **909**EB and **909**EC. These include Token Resources, Tagger Resources and Grouper Resources respectively. After initialization, the resources are freed. The memory that was used to store all Noun Phrases are also de-allocated.

Additional Embodiments

In an e-commerce embodiment of the present invention as illustrated in FIG. **13**, a web page **1300** contains typical visible links such as Books **1310**, Music **1320** so that on clicking the appropriate link the customer is taken to those pages. The web page may be implemented using HTML, a Java applet, or similar coding techniques which interact with the user's browser. For example, if customer wants to buy an album C by Artist Albert, he traverses several web pages as follows: he first clicks on Music (FIG. **13**, **1360**), which brings up page **1400** where he/she then clicks on Records (FIG. **14**, **1450**). Alternatively, he/she could select CDs **1460**, Videos **1470**, or other categories of books **1410**, music **1420** or help **1430**. As illustrated in FIG. **15**, this brings up another web page **1500** with links for Records **1550**, with sub-categories—Artist **1560**, Song **1570**, Title **1580**, Genre **1590**. The customer must then click on Artist **1560** to select the artist of choice. This displays another web page **1600** as illustrated in FIG. **16**. On this page the various artists **1650** are listed as illustrated—Albert **1650**, Brooks **1660**, Charlie **1670**, Whyte **1690** are listed under the category Artists **1650**. The customer must now click on Albert **1660** to view the albums available for Albert. When this is done, another web page is displayed as shown in FIG. **17**. Again this web page **1700** displays a similar look and feel, but with the albums available **1760**, **1770**, **1780** listed under the heading Titles **1750**. The customer can also read additional information **1790** for each album. This album information is similar to the liner notes of a shrink-wrapped album purchased at a retail store. One Album A is identified, the customer must click on the Album A **1760**. This typically brings up another text box with the information about its availability, price, shipping and handling charges etc.

When web page **1300** is provided with functionality of a NLQS of the type described above, the web page interacts with the client side and server side speech recognition modules described above. In this case, the user initiates an inquiry by simply clicking on a button designated Contact Me for Help **1480** (this can be a link button on the screen, or a key on the keyboard for example) and is then told by character **1440** about how to elicit the information required. If the user wants Album A by artist Albert, the user could articulate "Is Album A by Brooks available?" in much the same way they would ask the question of a human clerk at a brick and mortar facility. Because of the rapid recognition performance of the present invention, the user's query would be answered in real-time by character **1440** speaking out the answer in the user's native language. If desired, a readable word balloon **1490** could also be displayed to see the character's answer and so that save/print options can also be implemented. Similar appropriate question/answer pairs for each page of the website can be constructed in accordance with the present teachings, so that the customer is provided with an environment that emulates a normal conversational human-like question and answer dialog for all aspects of the web site. Character **1440** can be adjusted and tailored according to the particular commercial application, or by the user's own preferences, etc. to have a particular voice style (man, woman, young, old, etc.) to enhance the customer's experience.

In a similar fashion, an articulated user query might be received as part of a conventional search engine query, to locate information of interest on the INTERNET in a similar manner as done with conventional text queries. If a reasonably close question/answer pair is not available at the server side (for instance, if it does not reach a certain confidence level as an appropriate match to the user's question) the user could be presented with the option of increasing the scope so that the query would then be presented simultaneously to one or more different NLEs across a number of servers, to improve the likelihood of finding an appropriate matching question/answer pair. Furthermore, if desired, more than one "match" could be found, in the same fashion that conventional search engines can return a number of potential "hits" corresponding to the user's query. For some such queries, of course, it is likely that real-time performance will not be possible (because of the disseminated and distributed processing) but the advantage presented by extensive supplemental question/answer database systems may be desirable for some users.

It is apparent as well that the NLQS of the present invention is very natural and saves much time for the user and the e-commerce operator as well. In an e-support embodiment, the customer can retrieve information quickly and efficiently, and without need for a live customer agent. For example, at a consumer computer system vendor related support site, a simple diagnostic page might be presented for the user, along with a visible support character to assist him/her. The user could then select items from a "symptoms" page (i.e., a "monitor" problem, a "keyboard" problem, a "printer" problem, etc.) simply by articulating such symptoms in response to prompting from the support char-

acter. Thereafter, the system will direct the user on a real-time basis to more specific sub-menus, potential solutions, etc. for the particular recognized complaint. The use of a programmable character thus allows the web site to be scaled to accommodate a large number of hits or customers without any corresponding need to increase the number of human resources and its attendant training issues.

As an additional embodiment, the searching for information on a particular web site may be accelerated with the use of the NLQS of the present invention. Additionally, a significant benefit is that the information is provided in a user-friendly manner through the natural interface of speech. The majority of web sites presently employ lists of frequently asked questions which the user typically wades item by item in order to obtain an answer to a question or issue. For example, as displayed in FIG. **13**, the customer clicks on Help **1330** to initiate the interface with a set of lists. Other options include computer related items at **1370** and frequently asked questions (FAQ) at **1380**.

As illustrated in FIG. **18**, a web site plan for typical web page is displayed. This illustrates the number of pages that have to be traversed in order to reach the list of Frequently-Asked Questions. Once at this page, the user has to scroll and manually identify the question that matches his/her query. This process is typically a laborious task and may or may not yield the information that answers the user's query. The present art for displaying this information is illustrated in FIG. **18**. This figure identifies how the information on a typical web site is organized: the Help link (FIG. **13**, **1330**) typically shown on the home page of the web page is illustrated shown on FIG. **18** as **1800**. Again referring to FIG. **18**, each sub-category of information is listed on a separate page. For example, **1810** lists sub-topics such as 'First Time Visitors', 'Search Tips', 'Ordering', 'Shipping', 'Your Account' etc. Other pages deal with 'Account information' **1860**, 'Rates and Policies' **1850** etc. Down another level, there are pages that deal exclusively with a sub-sub topics on a specific page such as 'First Time Visitors' **1960**, 'Frequently Asked Questions' **1950**, 'Safe Shopping Guarantee' **1940**, etc. So if a customer has a query that is best answered by going to the Frequently Asked Questions link, he or she has to traverse three levels of busy and cluttered screen pages to get to the Frequently Asked Questions page **1950**. Typically, there are many lists of questions **1980** that have to be manually scrolled through. While scrolling visually, the customer then has to visually and mentally match his or her question with each listed question. If a possible match is sighted, then that question is clicked and the answer then appears in text form which then is read.

In contrast, the process of obtaining an answer to a question using a web page enabled with the present NLQS can be achieved much less laboriously and efficiently. The user would articulate the word "Help" (FIG. **13**, **1330**). This would immediately cause a character (FIG. **13**, **1340**) to appear with the friendly response "May I be of assistance. Please state your question?". Once the customer states the question, the character would then perform an animation or reply "Thank you, I will be back with the answer soon". After a short period time (preferably not exceeding 5-7 seconds) the character would then speak out the answer to the user's question. As illustrated in FIG. **18** the answer would be the answer **1990** returned to the user in the form of speech is the answer that is paired with the question **1950**. For example, the answer **1990**: "We accept Visa, MasterCard and Discover credit cards", would be the response to the query **2000** "What forms of payments do you accept?"

Another embodiment of the invention is illustrated in FIG. **12**. This web page illustrates a typical website that employs NLQS in a web-based learning environment. As illustrated in FIG. **12**, the web page in browser **1200**, is divided into two or more frames. A character **1210** in the likeness of an instructor is available on the screen and appears when the student initiates the query mode either by speaking the word "Help" into a microphone (FIG. **2**B, **215**) or by clicking on the link 'Click to Speak' (FIG. **12**, **1280**). Character **1210** would then prompt the student to select a course **1220** from the drop down list **1230**. If the user selects the course 'CPlusPlus', the character would then confirm verbally that the course "CPlusPlus" was selected. The character would then direct the student to make the next selection from the drop-down list **1250** that contains the selections for the chapters **1240** from which questions are available. Again, after the student makes the selection, the character **1210** confirms the selection by speaking. Next character **1210** prompts the student to select 'Section' **1260** of the chapter from which questions are available from the drop down list **1270**. Again, after the student makes the selection, character **1210** confirms the selection by articulating the 'Section' **1260** chosen. As a prompt to the student, a list of possible questions appear in the list box **1291**. In addition, tips **1290** for using the system are displayed. Once the selections are all made, the student is prompted by the character to ask the question as follows: "Please ask your query now". The student then speaks his query and after a short period of time, the character responds with the answer preceded by the question as follows: "The answer to your question . . . is as follows: . . . ". This procedure allows the student to quickly retrieve answers to questions about any section of the course and replaces the tedium of consulting books, and references or indices. In short, it is can serve a number of uses from being a virtual teacher answering questions on-the-fly or a flash card substitute.

From preliminary data available to the inventors, it is estimate that the system can easily accommodate 100-250 question/answer pairs while still achieving a real-time feel and appearance to the user (i.e., less than 10 seconds of latency, not counting transmission) using the above described structures and methods. It is expected, of course, that these figures will improve as additional processing speed becomes available, and routine optimizations are employed to the various components noted for each particular environment.

Again, the above are merely illustrative of the many possible applications of the present invention, and it is expected that many more web-based enterprises, as well as other consumer applications (such as intelligent, interactive toys) can utilize the present teachings. Although the present invention has been described in terms of a preferred embodiment, it will be apparent to those skilled in the art that many alterations and modifications may be made to such embodiments without departing from the teachings of the present invention. It will also be apparent to those skilled in the art that many aspects of the present discussion have been simplified to give appropriate weight and focus to the more germane aspects of the present invention. The microcode and software routines executed to effectuate the inventive methods may be embodied in various forms, including in a permanent magnetic media, a non-volatile ROM, a CD-ROM, or any other suitable machine-readable format. Accordingly, it is intended that the all such alterations and modifications be included within the scope and spirit of the invention as defined by the following claims.

What is claimed is:

1. A method of interacting with a user of a natural language query system comprising the steps of:

a) providing a speech recognition engine adapted to recognize a first set of words and/or phrases from a user during an interactive speech session;

wherein said first set of words and/or phrases can include a natural language query presented as continuous natural language spoken data;

b) providing a database of query/answer pairs concerning one or more topics which can be responded to by the natural language query system during said interactive speech based session with a user;

c) providing a natural language routine adapted to process said first set of words and/or phrases and identify a response to said natural language query based on said query/answer pairs;

wherein said natural language routine is adapted to consider only a subset of said first set of words and/or phrases, and further can consider words and/or phrases in said natural language query which are not present in said query/answer pairs to determine said response;

d) providing an interactive electronic agent coupled to said natural language routine and configured to:

i. provide a prompt to the user during said interactive speech based session with suggestions on queries which can be made to the natural language query system;

ii. provide a confirmation of a substance of said natural language query;

iii. provide said response to the user from the natural language query routine.

2. The method of claim 1, wherein the interactive electronic agent is further configured to provide feedback while said natural language routine is processing said user natural language query.

3. The method of claim 2, wherein at least one of said prompt, confirmation or response is processed by a text to speech engine and rendered into audible form for the user by the interactive electronic agent.

4. The method of claim 2, wherein the interactive electronic agent automatically communicates in a voice with speech characteristics based on a native language used by the user.

5. The method of claim 1, wherein the interactive electronic agent is incorporated as part of an e-commerce Web page and presented within a Web browser program operating on a client device operated by the user.

6. The method of claim 5 wherein the electronic interactive agent form is an animated character on a screen of the client device.

7. The method of claim 1 further including a step:

configuring perception related parameters of the electronic interactive agent based on preferences of said user, including one of a gender, a visual appearance, and/or voice characteristics including one of pitch, volume and/or speed.

8. The method of claim 1 wherein said subset of words and/or phrases in said natural language query can be assigned different weightings determined by said natural language routine.

9. The method of claim 1, wherein said speech recognition engine is distributed between a client device and a server system which receives streaming speech data having reduced latency data content before silence is detected and the utterance is complete.

10. The method of claim 1, wherein said response is selected based on comparing which of said subset of words and/or phrases in said speech-based query also appear in said query/answer pairs.

11. The method of claim 10, wherein a single response is selected based on determining which of said query/answer pairs has an optimal number of words and/or phrases overlapping with said natural language query.

12. The method of claim 1 further including providing a second natural language routine at a separate computing system adapted to assist with a response to said natural language query, such that said natural language query is recognized by said speech recognition engine, and multiple databases at different server systems can be considered in responding to natural language queries for a set of topic entries.

13. The method of claim 1 wherein the interactive electronic agent provides responses adjusted for a context experienced by the user.

14. The system of claim 1 wherein the interactive electronic agent provides responses adjusted for a context experienced by the user.

15. An electronic agent based natural language query system comprising:

a) a database of query/answer pairs concerning one or more topics which can be responded to by the natural language query system during an interactive speech based session with a user;

b) a speech recognition engine executing at a computing system and adapted to recognize a first set of words and/or phrases from a user during said interactive speech session;

wherein said first set of words and/or phrases can include a natural language query presented as continuous natural language spoken data;

c) a natural language routine executing at said computing system and adapted to process said words and/or phrases and identify a response to said natural language query derived from said query/answer pairs;

wherein said natural language routine is adapted to consider only a subset of said words and/or phrases, and can consider words and/or phrases in said natural language query which are not present in said query/answer pairs to determine said response;

d) an interactive agent routine coupled to said natural language routine and for providing an interactive electronic agent configured and controlled by one or more additional second routines to:

i. provide a prompt to the user during said interactive speech based session with suggestions on queries which can be made to the natural language query system;

ii. provide a confirmation of a substance of said natural language query;

iii. provide said response to the user from the natural language routine.

16. The system of claim 15, wherein the interactive electronic agent is further configured to provide feedback while said natural language routine is processing said user natural language query.

17. The system of claim 15, wherein the interactive electronic agent is incorporated as part of an e-commerce Web page and presented within a Web browser program operating on a client device operated by the user.

18. The system of claim 17 wherein the electronic interactive agent is an animated character on a screen of the client device.

19. The system of claim 15 further including a routine to configure perception related parameters of the electronic interactive agent, including one of a gender, a visual appearance, and/or voice characteristics including one of pitch, volume and/or speed.

20. The system of claim 15, wherein at least one of said prompt, confirmation or response is processed by a text to speech engine and rendered into audible form for the user by the interactive electronic agent.

21. The system of claim 15, wherein the interactive electronic agent automatically communicates in a voice with speech characteristics based on a native language used by the user.

22. The system of claim 15 wherein said subset of words and/or phrases in said natural language query can be assigned different weightings determined by said natural language routine.

23. The system of claim 15, wherein said speech recognition engine is distributed between a client device and a server system which receives streaming speech data having reduced latency data content before silence is detected and the utterance is complete.

24. The system of claim 15, wherein said response is selected based on comparing which of said subset of words and/or phrases in said speech-based query also appear in said query/answer pairs.

25. The system of claim 15, wherein a single response is selected based on determining which of said query/answer pairs has an optimal number of words and/or phrases overlapping with said natural language query.

26. The system of claim 15 further including providing a second natural language routine at a separate computing system adapted to assist with a response to said natural language query, such that said natural language query is recognized by said speech recognition engine, and multiple databases at different server systems can be considered in responding to natural language queries for a set of topic entries.

27. A method of interacting with a user of a client-server based natural language query system comprising the steps of:

a) initiating a first interactive speech session between a client device and a server system on behalf of a user;

b) providing a database of query/answer pairs coupled to the server system concerning one or more topics which can be responded to by the natural language query system during said first interactive speech based session with said user;

c) providing a speech recognition engine distributed between the client device and server system, and such that speech data is configured for said first interactive speech session to have a reduced data content to reduce latency, and is streamed to the server before silence is detected;

wherein said speech recognition engine is adapted to recognize a first set of words and/or phrases received from said client device from a user during said first interactive speech session concerning a first topic taken from said one or more topics;

further wherein said first set of words and/or phrases can include a natural language query presented as continuous natural language spoken data;

c) providing a natural language routine coupled to the server system adapted to process said first set of words and/or phrases and identify a response to said natural language query based on said query/answer pairs;

wherein said natural language routine is adapted to consider only a subset of said first set of word and/or phrases, and further can consider words and/or phrases in said natural language query which are not present in said query/answer pairs to determine said response;

d) providing an interactive electronic agent coupled to said natural language routine which is customized for the user and configured to:

i. provide a prompt to the user during said first interactive speech based session with suggestions on queries which can be made to the natural language query system concerning said first topic;

ii. provide a confirmation of a substance of said natural language query;

iii. provide said response to the user from the natural language query routine;

wherein said interactive electronic control agent is configured to provide iterative audible responses to user natural language queries directed to said one or more topics until said interactive speech based session is terminated.

28. The method of claim 27 further including a step: customizing perception related parameters of the electronic interactive agent, including one of a gender, a visual appearance, and/or voice characteristics including one of pitch, volume and/or speed.

29. The method of claim 27, wherein the electronic interactive agent can conduct a second interactive speech based session with said user directed to continue with additional queries and answers concerning said first topic.

* * * * *